UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Barcelona Est

TREBALL FI DE GRAU

Grau en Enginyeria Electrònica Industrial i Automàtica

DETECTORS I DESCRIPTORS DE PUNTS CARACTERÍSTICS EN IMATGES. UN ESTUDI COMPARATIU

Memòria i Annexos

Autor: Guillem Armenteras Bosom
Director: Antoni Grau Saldes

Convocatòria: Juny de 2019

Resum

L'objectiu del present projecte és realitzar una comparació dels principals algorismes de detecció i descripció de punts característics en imatges i la creació d'una aplicació pel reconeixement d'objectes utilitzant els anteriors.

El projecte comença exposant una documentació detallada de tals algorismes perquè qualsevol lector, amb uns coneixements matemàtics i de visió per computador mínims, sigui capaç d'entendre els algorismes i la seva posterior comparació.

Seguidament, s'exposa la comparativa dels algorismes en qüestió sota diferents transformacions utilitzant les implementacions que proporciona la llibreria "Open CV" per a Phyton.

Per acabar, s'explica una metodologia pel reconeixement d'objectes utilitzant els algorismes de detecció i descripció comparats. Aquesta s'utilitzarà per a la creació d'una aplicació la qual realitzi el reconeixement d'objectes.

Resumen

El objetivo del presente proyecto es realizar una comparación de los principales algoritmos de detección y descripción de puntos característicos en imágenes y la creación de una aplicación para el reconocimiento de objetos utilizando los anteriores.

El proyecto comienza exponiendo una documentación detallada de tales algoritmos para que cualquier lector, con unos conocimientos matemáticos y de visión por computador mínimos, sea capaz de entender los algoritmos y su posterior comparación.

Seguidamente, se expone la comparación de los algoritmos en cuestión utilizando las implementaciones que proporciona la librería “Open CV” para Phyton.

Por último, se explica una metodología para el reconocimiento de objetos utilizando los algoritmos de detección y descripción comparados. Esta se utilizará para la creación de una aplicación la cual realice el reconocimiento de objetos.

Abstract

The aim of this project is to make a comparison of the main feature detection and description algorithms and the creation of an application that performs object recognition.

The project begins exposing a detailed documentation of such algorithms so that any reader, with a minimum knowledge of mathematical and computer vision, could be able to understand them.

It is followed by a comparison of such algorithms using the implementations provided by the "Open CV" library for Phyton.

Finally, a methodology for object recognition is explained and used to create an application that performs object recognition.

Agraïments

Vull expressar el meu agraïment a totes aquelles persones i institucions que han fet possible la realització d'aquest treball.

En primer lloc, agrair al departament d'assajos climàtics d'Applus per a tot el suport rebut i sobretot per la flexibilitat d'horaris que m'han permès per a poder fer el seguiment amb el tutor i la realització del treball.

Per altre banda, agrair també el suport rebut des de la universitat per part del director del projecte, Antoni Grau, que ha ajudat a encaminar el projecte en tot moment.

Per acabar, vull agrair molt especialment el suport a totes les persones que han estat al meu costat durant la realització del treball: a la família i amics.

Glossari

- LoG: *Laplacian of Gaussian.*
- DoG: *Difference of Gaussians.*
- FAST: Algorimse anomenat "*Features from Accelerated Segment Test"*.
- BRIEF: Algorimse anomenat "*Binary Robust Independent Elementary Features"*.
- SIFT: Algorimse anomenat "*Scale-Invariant Feature Transform".*
- SURF: Algorimse anomenat "*Speeded up robust features".*
- SURF128: Versió extensa (de 128 dimensions) del algorisme anomenat "*Speeded up robust features".*
- ORB: Algorimse anomenat "*Oriented fast and Rotated Brief".*
- BF: Algorisme d'emparellament anomenat "*Brute force".*
- FLANN: Algorisme d'emparellament "*Fast Library for Approximate Nearest Neighbors".*
- RANSAC: Algorisme iteratiu per calcular els paràmetres d'un model matemàtic anomenat "*Random Sample Cosenus".*

Índex

Índex de figures

Índex de taules

1. Introducció

1.1. Objectius i abast

Com el nom del projecte indica, l'objectiu principal d'aquest projecte és realitzar una comparació dels principals algorismes de detecció i descripció de punts característics en imatges sota diferents transformacions. També s'ha proposat com a objectiu el disseny i elaboració d'una aplicació pel reconeixement d'objectes. Per assolir tals objectius globals, lògicament, se'n han hagut de marcar de més específics. Aquest es poden enumerar de la següent manera:

1. Elaboració d'una documentació detallada del funcionament dels algorismes utilitzats: Per a poder realitzar una comparació dels algorismes de detecció i descripció, ha sigut necessari realitzar un estudi de l'estat de l'art per a, entre d'altres, determinar quins d'aquests són els més genèrics, populars i utilitzats. S'ha determinat que aquests podrien ser SIFT, SURF i ORB, per tal motiu, es realitzarà la comparació de tals. Seguidament, s'ha elaborat una documentació que expliqui en detall el seu funcionament. També cal destacar que s'explicaran breument els algorismes d'emparellament de descriptors.
2. Creació d'un conjunt d'imatges adequat per a la posterior comparació: Com s'ha dit anteriorment, la comparativa es realitzarà enfront de diverses transformacions. Per aquest motiu ha sigut necessari crear un conjunt d'imatges amb tals transformacions.
3. Obtenció de propietats: Un cop es disposen de les imatges, s'han d'executar els diferents algorismes de detecció, descripció i emparellament sobre el conjunt d'imatges esmentat anteriorment. Lògicament, apart d'executar-los, s'han d'obtenir diferents propietats i resultats per a posteriorment avaluar-los.
4. Comparativa: Un cop es tenen les dades del funcionament dels algorismes sota diferents transformacions, aquestes han de ser analitzades per a arribar a certes conclusions.
5. Aplicació pel reconeixement d'objectes: S'ha proposat i explicat una metodologia pel reconeixement d'objectes utilitzant els algorismes de detecció i descripció comparats. Aquesta s'utilitzarà per a la creació d'una aplicació la qual realitzi el reconeixement d'objectes, la qual permeti a l'usuari utilitzar els diferents algorismes i canviar certs paràmetres d'aquests.

1.2. Software utilitzat

Per a assolir els objectius proposats anteriorment, s'ha hagut de escollir un entorn de programació i una llibreria de visió per computador la qual proporcioni tots els algorismes que es necessiten.

Després d'una exhaustiva recerca, s'ha determinat que la llibreria més adequada per a la realització d'aquest treball és "Open CV" (*Open Computer Vision*). Aquesta es caracteritza per ser de codi lliure, altament optimitzada, enfocada a aplicacions en temps real, i multi plataforma (pot ser utilitzada amb C++ i Phyton). Cal destacar també, que aquesta disposa d'una excel·lent documentació.

Seguidament, és necessari escollir el llenguatge. Després d'analitzar les possibles opcions s'ha escollit Phyton. Al ser un llenguatge d'alt nivell, és més lent que l'altre opció, C++, però les tasques més costoses del diferent codi que s'ha elaborat al llarg del projecte són els algorismes que proporciona Open CV, els quals són implementats amb C++. Per tal motiu, la diferència en la rapidesa entre Phyton i C++ en aquest cas és mínima. Cal remarcar que no es tenia coneixement d'aquest llenguatge, i les primeres setmanes de la realització del projecte es van dedicar a aprendre el funcionament d'aquest.

Finalment, per a la creació de l'aplicació o GUI (*Graphical User Interface)* s'ha escollit la llibreria estàndard i més utilitzada per a la creació d'aquestes en Phyton, anomenada Tkinter.

A continuació, es mostren les versions utilitzades del llenguatge i les llibreries utilitzades al llarg del projecte:

Llenguatge / Llibreria	Versió
Phyton	3.7.2
Open CV	2.4.6
Tkinter	8.6

Taula 1. Versions del llenguatge de programació i llibreries utilitzades.

2. Detectors i descriptors de punts característics

2.1. Introducció

En diferents aplicacions de visió per computador és necessari extreure certs punts característics i distintius d'una imatge i establir emparellaments amb els seus punts corresponents presents en una altre imatge i escena. Aquest procés consisteix en tres tasques: la detecció, la descripció i l'emparellament.

Considerant la Figura 2.1.1**,** la qüestió és, quins punts o regions s'haurien de detectar per a establir correspondències o connexions entre les dos imatges?

Figura 2.1.1 Dos parells d'imatges a emparellar. (Font: [6])

Intuïtivament, els primers punts que ens venen a la ment podrien ser els pics de les muntanyes, les cantonades de la casa o de les seves finestres, o potser algunes formes del relleu de la neu. Es trien aquests punts, perquè són distintius i únics en les imatges. Aquests punts o regions són anomenats punts d'interès (a vegades cantonades) i es troben amb els algorismes de detecció.

Un cop s'han detectat aquests punts d'interès, normalment són descrits amb l'aparença d'una regió de píxels veïna. Aquest procés el realitzen els algorismes de descripció, els quals, usualment, proporcionen un vector el qual descriu la regió veïna de cada punt d'interès.

Finalment, s'han d'emparellar els descriptors per a trobar les connexions entre les imatges. Els algorismes que realitzen aquesta funció són anomenats *feature matchers* o mètodes d'emparellament i s'explicaran detingudament a l'apartat 2.7.

2.2. Conceptes previs

És necessari introduir els següents conceptes i algorismes per a facilitar la comprensió dels algorismes de detecció i descripció, explicats posteriorment.

2.2.1. *Scale Space*

Primerament cal introduir el concepte d'escala en una imatge. Per dur-ho a terme es considera la següent imatge:

Figura 2.2.1. Quadre on s'aprecien diferents escales. (Font: [6])

El quadre anterior conté molts objectes o entitats a un gran rang d'escales, des d'objectes a petita escala com les rajoles del terra o els detalls en l'arquitectura, fins a estructures a gran escala com l'edifici.

A els humans no ens confon l'existència de múltiples escala en una escena, podem distingir perfectament l'edifici del quadre tot i que contingui multitud de detalls i components a una escala molt inferior, però en visió per computador no és immediat. La teoria anomenada *scale space* és utilitzada per a donar una representació a múltiples escales d'una senyal o imatge, d'aquesta manera es poden realitzar els anàlisis requerits a les diferents escales de la imatge.

Considerant una imatge $I(x, y)$, la seva representació a l'espai escalar en funció del paràmetre d'escala t és un conjunt de imatges $L(x, y, t)$ obtingudes mitjançant la convolució de la imatge amb de la *kernel* Gaussià de dos dimensions, $G(x, y, t)$.

$$L\,(x, y, t)\; =\; G\,(x, y, t) * \; I\,(x, y) \tag{2.2.1}$$

On:

$$G\,(x, y, t) = \frac{1}{2\pi t} e^{-\frac{(x^2+y^2)}{2t}} \tag{2.2.2}$$

A continuació es poden veure les representacions en l'espai escalar d'una imatge i d'una senyal unidimensional.

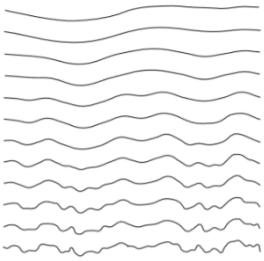


Figura 2.2.2. Representació en l'*scale space* d'una senyal unidimensional. (Font: [6])

$L\,(x, y, t), t = 0$ $L\,(x, y, t), t = 1$ $L\,(x, y, t), t = 4$

Figura 2.2.3. Representació en l'*scale space* d'una imatge. (Font: Elaboració pròpia)

El paràmetre d'escala t equival a σ^2, on σ és la desviació estàndard del *kernel* Gaussià. Per aquest motiu també es sol representar amb σ com a paràmetre d'escala, sent:

$$L\,(x, y, \sigma)\; =\; G\,(x, y, \sigma) * \; I\,(x, y) \tag{2.2.3}$$

On:

$$G\,(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}} \tag{2.2.4}$$

2.2.2. *Laplacian of Gaussian*

L'anomenada *Laplacian of Gaussian* s'obté al aplicar l'operador Laplacià ∇^2 a una imatge a escala σ de la representació en l'*scale space* $L\ (x, y, \sigma)$ de la imatge $I\ (x, y)$. Tal que:

$$\nabla^2 L = L_{xx} + L_{yy} \tag{2.2.5}$$

On L_{xx} i L_{yy} són les segones derivades en les corresponents direccions.

La imatge resultant té una gran resposta per a regions (*blobs)* de radi $r = \sqrt{2\sigma^2}$. El principal problema d'aquest operador és que la seva resposta té una gran dependència a la relació entre la mida de la regió i la desviació tipus aplicada al generar la representació en l'espai escalar. Per aquest motiu, tal i com es veurà més endavant, convé obtenir-la i analitzar-la a diferents escales .

A continuació, es mostren diverses *Laplacian of Gaussian* amb diferents σ d'una mateixa imatge:

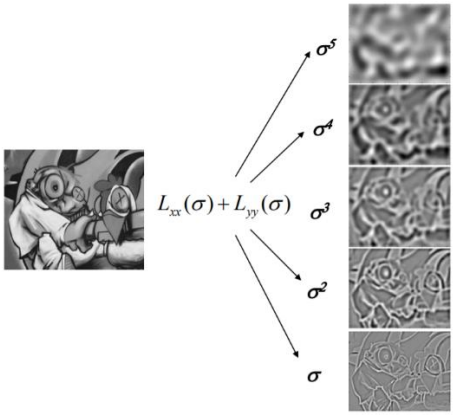


Figura 2.2.4. *Laplacian of Gaussian* amb diferents $\boldsymbol{\sigma}$. (Font: [9])

Aquest operador és utilitzat en diferents algorismes de detecció de punts característics.

2.2.3. Detector Harris Corner

El detector *Harris Corner* es basa en l'anàlisi de valors i vectors propis matriu de la matriu M, anomenada matriu de auto-correlació, definida com:

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x & I_x I_y \\ I_y I_x & I_y \end{bmatrix} \tag{2.2.6}$$

On I_x i I_y són les derivades en les respectives direccions i $w(x, y)$ és una finestra gaussiana o rectangular que s'utilitza per a seleccionar l'àrea d'interès, tal i com es veu a continuació:

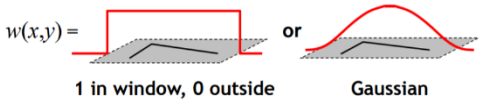


Figura 2.2.5. Finestra rectangular i gaussiana. (Font: [9])

Fent l'anàlisi de la matriu M es determinen les direccions i magnituds de màxim canvi en la regió considerada. Els valors propis (λ_0 i λ_1) en determinen les magnituds i els vectors les direccions. A continuació es presenta una representació gràfica:

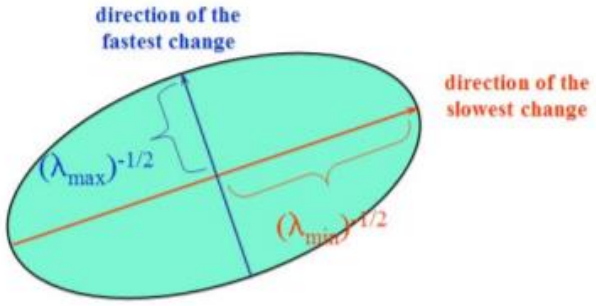


Figura 2.2.6. Direccions i magnituds de màxim i mínim canvi d'un punt o regió en funció dels valors i vectors propis. (Font: [6])

Definint la funció R tal que:

$$R = det(M) - \alpha\, trace(M)^2 = \lambda_0\lambda_1 + \alpha(\lambda_0 + \lambda_1) \qquad (2.2.7)$$

On α és una constant utilitzada per a donar més o menys importància a la suma de valors propis.

Analitzant la funció anterior es pot fer la següent classificació:

- Si $|R|$ és petit, el qual passa quan els dos valors propis són petits, la regió és plana (*flat*).
- Si $R < 0$, el qual passa quan un valor propi és molt major a l'altre, la regió és una línia (*edge*).
- Si R és gran, el qual passa quan els dos valors propis són grans i similars, la regió és una cantonada (*corner)*.

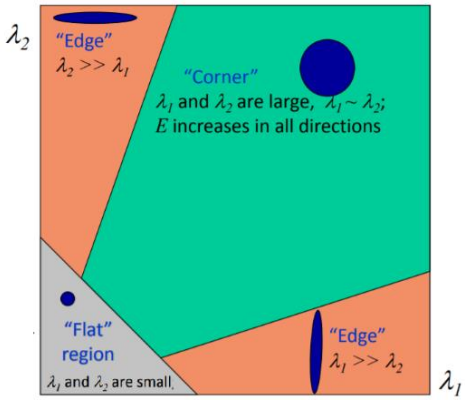


Figura 2.2.7. Classificació dels punts o regions en funció dels valors propis. (Font: [9])

Seguint amb la classificació anterior, òbviament els punts més característics són les cantonades, així que el detector *Harris* Corner fixa un llindar, calcula la resposta de R per tots els píxels i els que tenen una resposta més elevada que el llindar són considerats cantonades o punts d'interès.

Aquest detector manca d'invariància d'escala, motiu pel qual no és comunament utilitzat. Tot i així, la funció R i l'anàlisi dels valors propis són utilitzats en diferents algorismes de detecció de més rellevància. Es remarca també es pot realitzar aquesta classificació en funció dels valors i vectors propis de la matriu Hessiana, definida en el pròxim apartat.

2.2.4. Detector Hessià

El detector Hessià es basa en el determinant de la matriu Hessiana, aquesta es defineix com:

$$H(x,y) = \begin{bmatrix} I_{xx}(x,y) & I_{xy}(x,y) \\ I_{yx}(x,y) & I_{yy}(x,y) \end{bmatrix} \tag{2.2.8}$$

On I_{xx} són les segones derivades en les respectives direccions.

El detector utilitza el determinant d'aquesta matriu, el qual té una forta resposta enfront a altes derivades en dos direccions ortogonals.

$$det(H) = I_{xx}I_{yy} - I_{xy}^2 \tag{2.2.9}$$

A continuació, es pot veure l'alta resposta a les cantonades al aplicar el determinant de la matriu Hessiana:

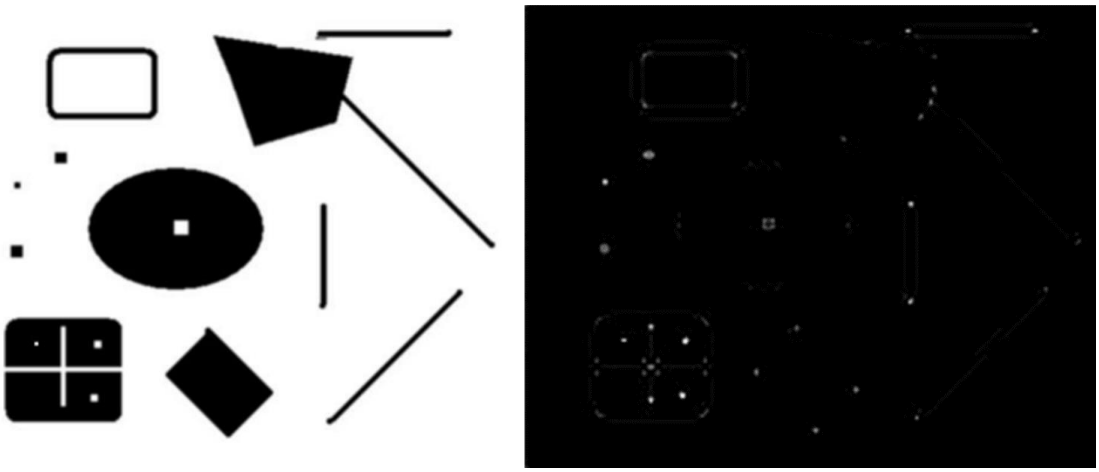


Figura 2.2.8. Imatge resultant al aplicar el determinant de la matriu Hessiana (dreta) en la imatge inicial (esquerra). (Font: [9])

Finalment, el detector Hessià fixa un llindar i tots els píxels amb valor del determinant major que aquest són considerats cantonades. L'algorisme de detecció *SURF* es basa el determinant de la matriu Hessiana, amb un seguit de millores.

2.2.5. Detector FAST

Edward Rosten i Tom Drummond van introduir un nou algorisme de detecció a l'article [3], anomenat *FAST (Features from Accelerated Segment Test)*. Aquest es caracteritza per la seva rapidesa, però manca d'invariància a l'escala i rotació.

L'algorisme opera considerant una circumferència de 16 píxels al voltant del píxel *p,* candidat a ser punt d'interès, tal i com s'observa a la següent figura:

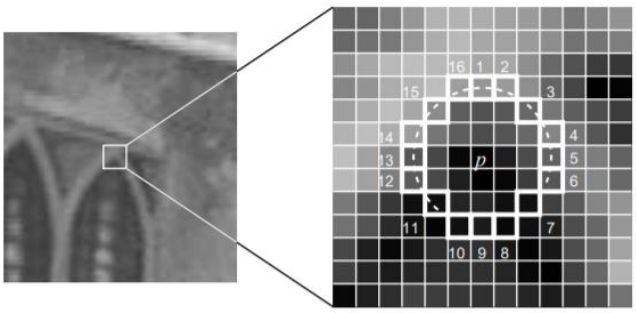


Figura 2.2.9. Circumferència dels 16 píxels utilitzats pel detector. (Font: [3])

El detector classifica el punt *p* com a punt d'interès sí es compleix el criteri que existeixi un conjunt de *n* píxels contigus els quals tinguin un valor de lluminositat major a la del píxel candidat, I_p , sumat d'un llindar *t, o* si tots tenen un valor menor a I_p - *t.*

El nombre de píxels contigus *n* es va escollir de 12 ja que, després d'un estudi, es va considerar que era el nombre òptim que preservava la qualitat i rapidesa del detector.

Per a augmentar la rapidesa d'execució, l'algorisme segueix els següents passos:

1. Es selecciona el píxel *p* i s'obté la seva intensitat I_p
2. Es compara la intensitat dels píxels 1,5,9 i 13 (marcats en la Figura 2.2.9) amb I_p seguint el criteri esmentat anteriorment. Com és obvi, com a mínim tres dels quatre píxels han de satisfer el criteri per a que el píxel *p* sigui candidat.
3. Si no es compleix el criteri en tres dels quatre píxels es rebutja *p* com a punt d'interès. Sinó, s'obté la intensitat dels 16 píxels i es comprova si 12 contigus el compleixen. Si és així, *p* és considerat punt d'interès
4. Repetir el procés per a tots els píxels de la imatge.

El detector en aquestes condicions presenta un alt rendiment computacional però detecta múltiples punts adjacents un a l'altre. Per a solucionar aquest problema s'utilitza el mètode que s'explicarà a continuació.

Com que el mètode anterior no calcula cap funció que retorni la resposta dels punts o cantonades, és necessari calcular una funció de puntuació V per a cada possible punt d'interès, definida com:

$$V = \max \sum_{x \in Sclars} \left|I_{p\to x} - I_p\right| - t \quad , \sum_{x \in Sfoscos} \left|I_{p\to x} - I_p\right| - t \tag{2.2.10}$$

On:

$$Sclars = \left\{x \,|I_{p\to x} \geq\ I_p + t\right\} \tag{2.2.11}$$

$$Sfoscos = \left\{x \,|I_{p\to x} \geq\ I_p - t\right\} \tag{2.2.12}$$

D'aquesta manera cada punt té una puntuació i el píxel que la tingui més elevada de *n* punts adjacents serà escollit finalment com a punt d'interès.

2.2.6. Descriptor BRIEF

FAST es sol utilitzar en aplicacions on la velocitat és clau. En aquestes, també es sol recórrer a descriptors d'alta rapidesa, com *BRIEF*.

Aquest algorisme proposa crear un vector de bits com a descriptor, el qual és altament discriminat i pot ser posteriorment aparellat utilitzant la distància *Hamming,* la qual és molt més ràpida de calcular en comparació a la distància euclidiana, la qual s'utilitza habitualment.

El descriptor simplement crea un vector bits en funció de comparacions d'intensitats entre parells de punts dins d'una regió al voltant dels punts d'interès (obtinguts anteriorment amb el detector).

Definint el test τ sobre la regió *p* de dimensions *SxS* com:

$$\tau(p;x,y) := \begin{cases} 1, & si\ I(x) < I(y) \\ x, & Altrament \end{cases} \quad (2.2.13)$$

On $I(x)$ és la intensitat del píxel al punt $x = (u, v)$ de la regió *p* suavitzada amb una gaussiana.

Per a produir el descriptor, és necessari escollir un conjunt de n_d (x, y) parells de localitzacions per a realitzar n_d tests i obtenir un vector de longitud n_d. D'aquesta manera es defineix el descriptor com:

$$f_{n_d}(p) = \sum_{1 \le i \le n_d} 2^{i-1}\ \tau(p; x_i, y_i) \quad (2.2.14)$$

L'article afirma que els valors òptims de n_d són 128, 256 i 512 pel què fa a la velocitat i eficiència. Amb aquests valors, els descriptors poden ser emmagatzemats amb 16, 32 i 64 bytes.

Un cop s'ha definit el descriptor, s'ha de decidir de quina manera suavitzar les regions i Necessitat de suavitzar

La construcció del mètode de test anterior és altament sensible al soroll ja que només pren informació de dos píxels. Aquesta sensibilitat es redueix suavitzant la regió amb un *kernel* gaussià, incrementant la repetibilitat i estabilitat de l'algorisme. Experimentalment s'ha trobat que el *kernel* gaussià amb variància de valor 2 proporciona bons resultats.

2.2.6.1. Localització dels parells de punts de test

Finalment s'ha de decidir com crear un conjunt localitzacions dels punts de test. Existeixen diferents opcions per generar un nombre de n_d localitzacions de test dins d'una regió de SxS píxels. A l'article [5] han experimentat amb 5 diferents geometries, les quals es poden veure a la següent figura.

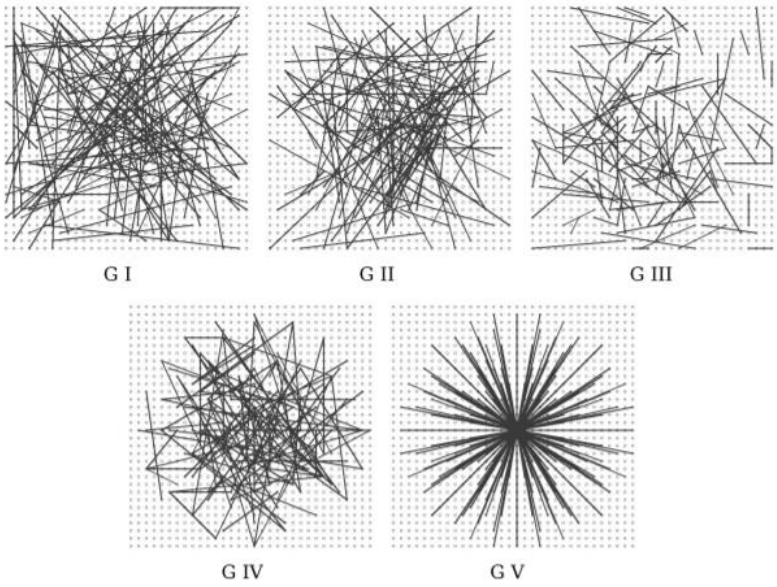


Figura 2.2.10. Diferents distribucions de les localitzacions de test. (Font: [5])

Aquestes són descrites com:

I. Les localitzacions x_i i y_i són generades aleatòriament amb una distribució uniforme.
II. Les localitzacions x_i i y_i són generades aleatòriament amb una distribució gaussiana amb $\sigma^2 = \frac{S^2}{25}$ centrada a l'origen.
III. Les localitzacions x_i són generades aleatòriament amb una distribució gaussiana amb $\sigma^2 = \frac{S^2}{25}$ centrada a l'origen. Seguidament, les y_i també són generades seguint una gaussiana però amb $\sigma^2 = \frac{S^2}{100}$ i centrades a x_i.
IV. x_i i y_i està ubicades a localitzacions discretes d'una quadrícula polar.
V. Els punts x_i estan ubicats a l'origen i els y_i ocupen tots els possibles punts d'una quadrícula polar de n_d punts.

A l'article [5] es va determinar experimentalment que la distribució espacial II és la que donava millors resultats.

2.3. SIFT

Al 2004, David Lowe va publicar l'article [1], on s'exposa l'algoritme anomenat *SIFT* (*Scale-invariant feature transform*). Aquest presenta un mètode per extreure i punts característics distintius i invariants i descriure'ls. L'algorisme es va dissenyar per tal que aquests siguin invariants a la rotació i escala. També obté un emparellament robust enfront a distorsions afins, canvis en el punt de vista 3D, addicció de soroll i canvis d'il·luminació.

2.3.1. Detector SIFT

2.3.1.1. Detecció d'extrems a l'*scale space*

Primerament, es detecten punts invariants a l'escala. Per dur-ho a terme es busquen punts estables en un conjunt d'escales utilitzant la funció de diferència de gaussianes per a obtenir una aproximació de la *Laplacian of Gaussian* (posteriorment es farà referència a aquesta funció com a *LoG*) a diferents escales**.** A l'article [2] es demostra que *LoG* pot ser aproximada amb la funció diferència de gaussianes.

Per a entendre la funció de diferència de gaussianes, primerament cal recordar la funció d'espai escalar d'una imatge, la qual ha set explicada als conceptes previs. Aquesta és definida com la funció $L\ (\mathrm{x}, \mathrm{y}, \sigma)$, obtinguda de la convolució amb de la variable escalar Gaussiana, $\mathrm{G}\ (\mathrm{x}, \mathrm{y}, \sigma)$, amb la imatge $\mathrm{I}\ (\mathrm{x}, \mathrm{y})$. S'expressa de la següent manera:

$$L\ (x, y, \sigma)\ =\ G\ (x, y, \sigma) * \ I\ (x, y) \tag{2.3.1}$$

On:

$$G\ (x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}} \tag{2.3.2}$$

Com s'ha dit anteriorment, l'algoritme obté una aproximació de la *LoG* amb la funció de diferència de gaussianes $D\ (\mathrm{x}, \mathrm{y}, \sigma)$, definida en (2.3.3), la qual pot ser obtinguda restant dos escales properes separades per una constant *k*, tal i com es veu en (2.3.4).

$$D\ (x, y, \sigma)\ = (G\ (x, y, k\sigma) - G\ (x, y, \sigma)) * \ I\ (x, y) \tag{2.3.3}$$

$$D\ (x, y, \sigma)\ = L\ (x, y, k\sigma) - \ L\ (x, y, \sigma) \tag{2.3.4}$$

El motiu pel qual SIFT utilitza aquesta funció per a trobar *LoG* és que computacionalment és molt més eficient.

A la Figura 2.3.1 es mostra la construcció eficient de $D\ (\mathrm{x}, \mathrm{y}, \sigma)$ que realitza l'algorisme.

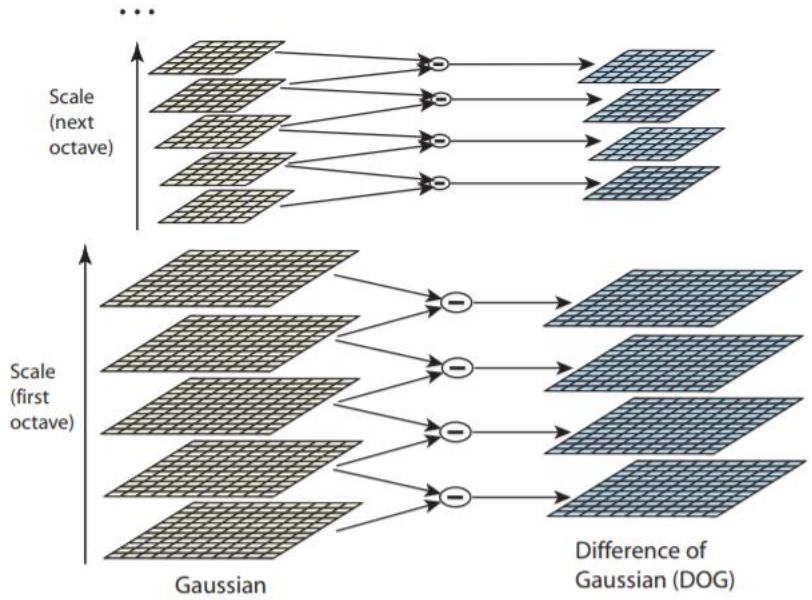


Figura 2.3.1. Construcció eficient del conjunt de diferències de gaussianes a diferents escales i octaves. (Font: [1])

Inicialment, es fa la convolució amb el *kernel* gaussià sobre la imatge inicial per produir imatges separades una constant *k* en l'espai escalar. Llavors, es resten les imatges adjacents per a obtenir el conjunt de diferències de gaussianes a les diferents escales. A l'article [1] es recomana el valor de $k = \sqrt{2}$ i un nombre de cinc escales per octava. De tal manera, l'escala més elevada haurà sigut convolucionada amb una σ doble que la inicial.

Un cop s'ha completat una octava, es submostreja la imatge que s'ha convolucionat amb un valor de σ doble que l'inicial (si es tenen cinc escales per octava, serà la imatge amb escala superior de l'anterior octava), treien cada segon píxel de totes les columnes i files. D'aquesta manera s'obté la primera imatge de la següent octava, amb una resolució igual a la meitat que la de la imatge inicial. Seguidament es repeteix el procés per la resta d'octaves. A l'article [1] es fa un estudi i determina que σ = 1,6 dona els millors resultats.

Per detectar els màxims i mínims locals de $D(\mathrm{x}, \mathrm{y}, \sigma)$, cada punt és comparat amb els 8 veïns de la pròpia imatge i amb els 9 de l'escala superior i inferior, tal i com es mostra en la Figura 2.3.2. Si el punt en qüestió és major o menor a tots els altres, aquest es considerat un punt de potencial interès.

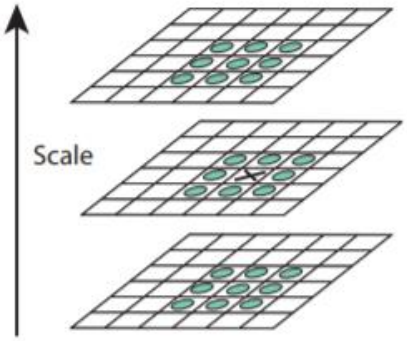


Figura 2.3.2. Comparació amb els 26 píxels veïns per a detecció de màxims i mínims locals. (Font: [1])

2.3.1.2. Localització precisa dels punts característics

Els punts obtinguts en l'anterior apartat són candidats a ser punts d'interès, però és possible que siguin punts amb poc contrast o que estiguin localitzats sobre una línia (recordem que la *Laplacian of Gaussian* té una forta resposta a línies i cantonades).

Per a descartar aquests punts i obtenir una major robustesa, es desenvolupa un mètode per obtenir els candidats més adequats. Per a aconseguir-ho, s'utilitza l'expansió amb series de Taylor de la funció $D\ (\mathrm{x}, \mathrm{y}, \sigma)$:

$$D\ (\mathrm{x}) = D + \frac{\partial D^T}{\partial \mathrm{x}}\ \mathrm{x} + \frac{1}{2}\mathrm{x}^T \frac{\partial^2 D^t}{\partial \mathrm{x}^2}\ \mathrm{x} \tag{2.3.5}$$

On $D\ (\mathrm{x})$ i les seves derivades són avaluades al punt d'interès $\mathrm{x} = (\mathrm{x}, \mathrm{y}, \sigma)^T$.

La localització de l'extrem $\hat{x}$ s'obté igualant-la a zero la derivada de la funció $D\ (\mathrm{x})$ respecte a x, donant:

$$\hat{x} = -\frac{\partial^2 D^{-1}}{\partial \mathrm{x}^2}\ \frac{\partial D}{\partial \mathrm{x}} \tag{2.3.6}$$

On les derivades són obtingudes mitjançant diferències entre veïns i el sistema lineal de 3 variables és resolt amb cost mínim.

Si el punt $\hat{x}$ queda situat a una distancia de x major a una meitat en qualsevol direcció, significa que el màxim o mínim queda més proper a un punt diferent. Llavors x es descarta i es fa una interpolació entre ambdós punts ($\hat{x}$ i x) per a aconseguir un nova localització i escala més precisa.

El valor de la funció a l'extrem $D\ (\hat{x})$ obtingut substituint l'equació (2.2.6) a (2.2.7) és útil per a rebutjar extrems inestables amb poc contrast.

$$D\ (\hat{x}) = D + \frac{1}{2}\frac{\partial\, D^T}{\partial \mathrm{x}}\ \hat{x} \tag{2.3.7}$$

Així doncs l'algorisme rebutja tots els extrems amb valors $|D\ (\hat{x})|$ menors a un cert valor, en l'article recomenen el valor de 0,03 (assumint que el rang de valors dels pixels és [0, 1]).

2.3.1.3. Eliminació de les *edge responses*

En l'apartat anterior s'han eliminat el punts amb poc contrast, ara convé eliminar els que queden localitzats sobre línies (*edges)*.

Com s'ha explicat en els conceptes previs¸ aquests punts tenen una gran principal curvatura en una direcció i una molt inferior en la direcció perpendicular. Les principals curvatures es poden trobar amb els valors propis de la matriu Hessiana 2x2 avaluada a la localització i escala del punt d'interès.

Es recorda que la matriu Hessiana es defineix, en aquest cas, com:

$$H(x, y, \sigma) = \begin{bmatrix} D_{xx}(x, y, \sigma) & D_{xy}(x, y, \sigma) \\ D_{yx}(x, y, \sigma) & D_{yy}(x, y, \sigma) \end{bmatrix} \tag{2.3.8}$$

Les derivades $D_{xx}(x, y, \sigma)$ són obtingudes a partir de les diferències entre veïns.

Els valors propis són proporcionals a les principals curvatures de D. Sent α el valor propi amb major magnitud i β el menor:

$$Tr(H) = D_{xx} + D_{yy} = \alpha + \beta \tag{2.3.9}$$

$$det(H) = D_{xx}D_{yy} - \left(D_{xy}\right)^2 = \alpha\beta \tag{2.3.10}$$

No ens interessa el seu valor, sinó si un és major que l'altre. D'aquesta manera, sent *r* el ràtio entre els valors propis:

$$\frac{Tr(H)^2}{Det(H)} = \frac{(\alpha + \beta)^2}{\alpha\beta} = \frac{(r\beta + \beta)^2}{r\beta^2} = \frac{(r+1)^2}{r} \tag{2.3.11}$$

L'expressió anterior només depèn del ràtio entre valors propis, i dona un mínim quan aquests són iguals i incrementa a mesura que incrementa el ràtio. Així doncs, per a comprovar que el ràtio és inferior a un cert valor, l'algorisme només ha de avaluar l'expressió (2.3.11), la qual és avaluada molt eficientment. A l'article es recomana prendre el valor *r* = 10.

2.3.1.4. Assignació d'orientació

A l'assignar una orientació a cada punt d'interès es pot crear un descriptor a on es representi la orientació relativa i aconseguir invariància a la rotació.

Com s'ha explicat anteriorment, cada punt d'interès és representat per les seves coordenades (x, y) i la seva escala σ. L'escala del punt és usada per a seleccionar la imatge de l'espai escalar $L(x, y, \sigma)$ amb la σ més pròxima. Seguidament, es calcula la magnitud del gradient $m(x, y)$ i la orientació $\theta(x, y)$ sobre la imatge seleccionada utilitzant la diferència de píxels, amb les següents expressions:

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2} \qquad (2.3.12)$$

$$\theta(x,y) = \tan^{-1}\left(\frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)}\right) \qquad (2.3.13)$$

Seguidament, es forma un histograma d'orientacions format amb els gradients i orientacions obtinguts d'una regió de veïns del punt d'interès (x, y). L'histograma té 36 orientacions, corresponents els 360º, i cada punt és ponderat amb la magnitud del gradient.

Els pics de l'histograma d'orientació corresponent a les direccions dominants dels gradients locals. Es detecta el pic més elevat i qualsevol altre que tingui un valor major al 80% d'aquest. Les direccions dels pics en qüestió són utilitzades per a crear un punt d'interès amb tal orientació. De tal manera, és possible que existeixin múltiples punt d'interès a una mateixa localització però amb diferents orientacions.

2.3.2. Descriptor SIFT

En aquest punt es disposa de els diferents punts d'interès, conformats per la seva localització en la imatge, l'escala i la orientació. El pròxim pas és construir un descriptor el màxim distintiu i alhora invariant a les possibles variacions, com un canvi d'il·luminació o un canvi del punt de vista.

La següent figura il·lustra el procés de computació del descriptor.

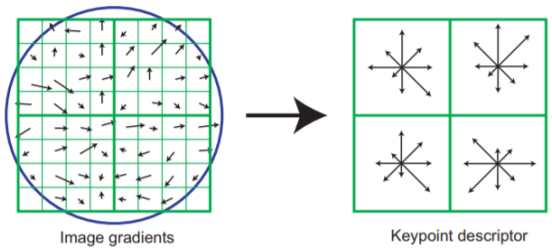


Figura 2.3.3. Creació del descriptor. (Font: [1])

Primerament es calculen les magnituds dels gradients i orientacions al voltant de la localització del punt d'interès tal com s'ha explicat a l'apartat anterior. Aquest procés es visualitza a l'esquerra de la Figura 2.3.3, on les magnituds i orientacions són representades amb les petites fletxes.

Llavors les magnituds són ponderades amb una gaussiana circular amb σ igual a la meitat de l'amplada del descriptor. Aquest procés l'il·lustra la circumferència blava de la part esquerra i és d'utilitat per donar més importància als punts més pròxims al punt d'interès.

A la dreta de la figura es mostra el descriptor. És permissiu a significants desplaçaments de les posicions dels gradients, ja que crea un histograma per cada regió de 4x4 píxels veïns. A cada histograma s'hi representen 8 direccions, amb la longitud de cada fletxa representant la magnitud de l'histograma.

El descriptor és format per un vector que conté els valors de totes les orientacions de l'histograma. La figura mostra una matriu 2x2, mentre que l'algorisme utilitza una matriu 4x4 d'histogrames amb 8 orientacions cada un. D'aquesta manera el descriptor té 128 elements per cada punt d'interès.

Per acabar, el vector és normalitzat. D'aquesta manera, al patir un canvi en la lluminositat de la imatge amb el qual cada píxel és multiplicat per una constant, també multiplicarà el gradient i es cancel·larà l'efecte degut a la normalització. Per altre banda, si el canvi de lluminositat és degut al sumar una constat a cada píxel, el gradient (calculat amb diferència de píxels) tampoc es veurà afectat. Així doncs, el descriptor és invariant a canvis lineals en la il·luminació.

Tot i així, existeixen canvis de lluminositat no-lineals deguts a la saturació de la càmera o a diferents reflexions en funció de la forma d'un objecte, entre d'altres. Aquests efectes solen causar un gran canvi al gradient i no tant a l'orientació. Per aquest motiu, es fixa un límit màxim de 0,2 als gradients i si un gradient té valor major se li estableix tal màxim. D'aquesta manera es dona més importància a la distribució de les orientacions que als gradients de gran valor.

2.4. SURF

Posteriorment a la creació del SIFT, es publicar l'article [2], el qual introduïa un nou algoritme anomenat SURF *(Speeded Up Robust Features)* per a la detecció i descripció de punts invariants a l'escala i a la rotació. Com el nom indica, aquest nou mètode es caracteritza per ser un mètode ràpid.

2.4.1. Detector SURF

El detector es basa en la matriu Hessiana, degut al seu bon rendiment en temps computacional. El determinant d'aquesta serà utilitzat per a la detecció de localitzacions i escales de punts d'interès. Donat un punt $x = (x, y)$ en una imatge I a escala σ, la matriu Hessiana $H(x, \sigma)$ es defineix com:

$$H(x, \sigma) = \begin{bmatrix} L_{xx}(x, \sigma) & L_{xy}(x, \sigma) \\ L_{yx}(x, \sigma) & L_{yy}(x, \sigma) \end{bmatrix}$$

On $L_{xx}(x, \sigma)$ és la convolució del kernel gaussià de segon ordre $\frac{\partial^2}{\partial x^2} g(\sigma)$ amb la imatge I.

Com s'ha esmentat anteriorment, és necessària la convolució amb gaussianes per a l'obtenció i anàlisi a l'espai escalar. Però a la pràctica, la gaussiana ha de ser discretitzada i retallada. Degut a aquesta no idealitat, SURF proposa l'alternativa d'aproximar $H(x, \sigma)$ amb els anomenats *box filters* (es mostren a la Figura 2.4.1) els quals són una aproximació de les derivades gaussianes de segon ordre. L'interessant és que aquestes poden ser avaluades molt ràpidament utilitzant imatges integrals.

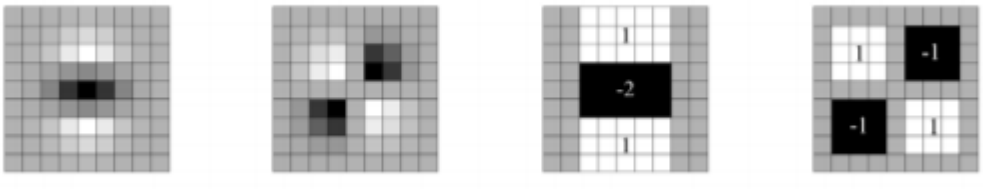


Figura 2.4.1. Esquerra: Derivades parcials de segon ordre en direccions *x* i *y* del *kernel* Gaussià. Dreta: *Box filters* utilitzats per a aproximar-los. (Font: [2])

Els filtres 9x9 que es veuen a la figura anterior són aproximacions de les derivades de segon ordre Gaussianes amb σ = 1.2, les quals representen la escala espacial més petita. Els pesos dels píxels d'aquests filtres són normalitzats respecte la mida del filtre, ja que es treballa amb diferents mides de filtre per a diferents escales.

Per a treballar amb diferents escales usualment s'implementen piràmides. Una imatge és repetidament suavitzada amb un filtre gaussià i posteriorment submostrejada per aconseguir nivells

superiors de la piràmide. Una alternativa a aquest mètode, el que utilitza *SURF*, consta en repetidament filtrar la imatge augmentant la mida del filtre. Al treballar amb els *box filters*, aquest procés es pot fer en paral·lel per a diferents escales i a gran velocitat gràcies a les imatges integrals.

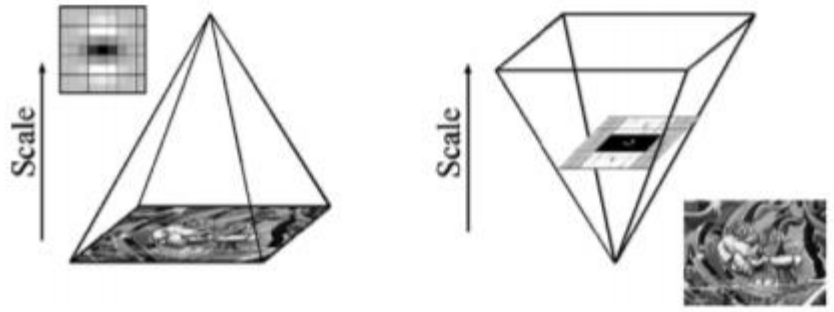


Figura 2.4.2. En comptes d'iterativament reduir la resolució de la imatge (esquerra), SURF augmenta la mida del *kernel* (dreta) . (Font: [2])

Així doncs, l'espai escalar és analitzat incrementant la mida del filtre. El resultat del filtre 9x9, vist en la Figura 2.4.2, és considerat com l'escala inicial (s'anomena escala *s*=1.2, fent referència a les derivades gaussianes amb σ = 1.2). Les següents escales són obtingudes amb filtres de mida 15×15, 21×21, 27×27, etc.

Per a localització punts d'interès sobre totes les escales s'utilitza el mateix mètode que SIFT (explicat en detall a l'apartat 2.3.1.2), però detectant els extrems locals del determinant de la matriu Hessiana en comptes dels de la funció diferència de gaussianes.

Recordem que primerament cada píxel és comparat a una regió de 3x3x3 píxels veïns per a l'obtenció de punts potencials. Seguidament s'obté una aproximació de la funció mitjançant l'expansió amb sèries de Taylor i amb aquesta es troba el màxim local. Si aquest màxim queda desplaçat del punt d'interès, es fa una interpolació per a obtenir l'escala i localització final del punt d'interès.

Finalment, es descarten els punts que, avaluats amb el determinant de la matriu Hessiana tinguin un valor inferior a un cert llindar.

2.4.2. Descriptor SURF

2.4.2.1. Assignació d'orientació.

Primerament, es filtren les diferents escales de la imatge amb els *kernels Har* wavelet en les direccions *x* i *y* (representats a la Figura 2.4.3) els quals aproximen les derivades corresponents**.** Llavors, per un punt d'interès que esta situat a l'escala *s*, es considera el veïnat circular de radi 6*s* de les respostes de *Har wavelet* a la mateixa escala*.* Aquest veïnat es pondera amb una gaussiana amb σ = 2*s* al voltant del punt. Seguidament, les respostes en *x* i *y* de tot el veïnat són representades com a vectors a un espai bidimensional on les abisses representen la resposta *x* i les ordenades la resposta *y.* Finalment, l'orientació dominant del punt és estimada calculant la suma de les respostes dins d'una finestra corredissa que cobreix un angle de 60 graus (sector circular grisós de la imatge Figura 2.4.3). La suma major dona resultat a la orientació del punt d'interès.

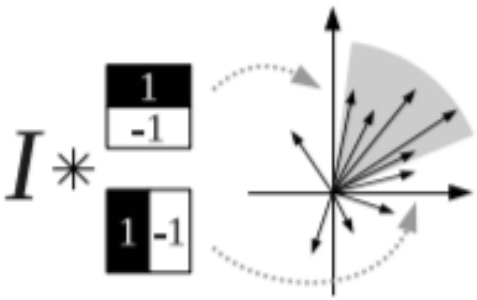


Figura 2.4.3. Representació del mètode d'obtenció d'orientació utilitzat en SURF. (Font: [2])

2.4.2.2. Components del descriptor

El primer pas per a extreure el descriptor és construir una regió quadrada centrada al voltant del punt d'interès i orientada a llarg de la orientació d'aquest, per a assolir invariància a la rotació.

La mida de la regió és 20*s* i aquesta és dividida a 4x4 sub-regions quadrades (16 sub-regions). Es consideren de nou les imatges filtrades amb els *kernels Haar wavelet* (a partir d'ara aquestes s'anomenaran d_x i d_y). Per augmentar la robustesa a transformacions geomètriques i soroll, d_x i d_y són ponderades amb una gaussiana (σ = 2*s*) centrada al punt d'interès.

Llavors, les respostes d_x i d_y de cada sub-regió són sumades i juntament amb la suma dels seus valors absoluts (per aportar informació de la polaritat dels canvis) es forma el següent vector, $v = (\sum d_x , \sum d_y , \sum |d_x| , \sum |d_y|)$. A continuació un exemple del vector v d'una subregió.

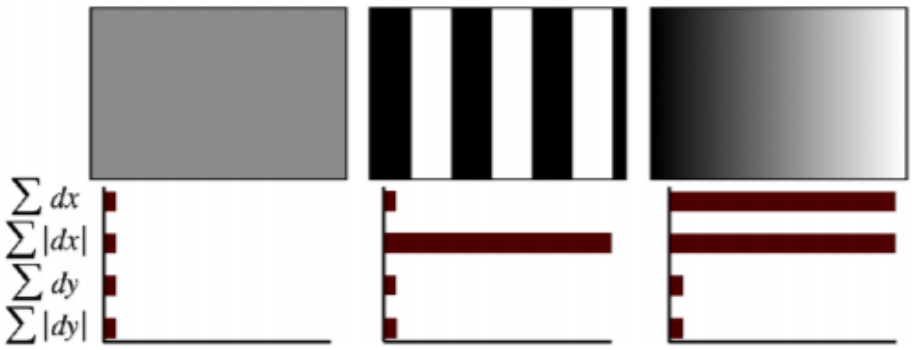


Figura 2.4.4. Components del descriptor SURF. En el cas d'una regió homogènia, totes els valors són relativament petits. En presència de freqüències en la direcció *x*, el valor $\sum |\boldsymbol{d_x}|$ és alt, mentre que els altres prenen valors baixos. Finalment, en un canvi gradual en la direcció *x*, tant $\sum |\boldsymbol{d_x}|$ com $\sum \boldsymbol{d_x}$ són alts. (Font: [2])

Finalment, el descriptor representa els vectors v de les 16 regions, formant un vector de 64 dimensions.

2.4.2.3. Descriptor extens

El descriptor té una versió extensa. Utilitza els mateixos valors que anteriorment, però aprofundint més. Les sumes de d_x i $|d_x|$ són calculades separadament per $d_y < 0$ i per $d_y \geq 0$. De mateixa manera, les sumes de d_x i $|d_x|$ són realitzades tenint en compte el signe de d_y.

Aquestes propietats formen un vector de 8 dimensions per a cada sub-regió, així que finalment el descriptor té 128 dimensions.

2.4.2.4. *Upright* SURF

Existeixen diverses aplicacions a les quals no es requereix invariància a la rotació, de manera que no és necessari trobar l'orientació ni orientar la regió per al procés de descripció. Per aquest motiu existeix la variant *Upright SURF*, la qual no realitza aquests passos, augmentant la velocitat de l'algorisme.

2.5. ORB

Al 2011 es va publicar l'article [6], anomenat: *"ORB: An efficient alternative to SIFT or SURF"*. Tal i com indica el títol, es presenta com a una bona alterativa a SIFT i a SURF tant en rendiment com en velocitat.

Com es veurà a continuació, *ORB* és bàsicament una agrupació del detector *FAST* i el descriptor *BRIEF*, amb diferents modificacions per a millorar-ne la qualitat i rendiment.

2.5.1. Detector FAST orientat

Com s'ha estudiat anteriorment (apartat 2.2.5), els punts obtinguts amb el *FAST* no tenen assignats cap component d'orientació, cosa que els fa poc robustos a la rotació. Arran d'això, ORB afegeix un mètode per a calcular-la eficientment. També s'aconsegueix una invariància a l'escala amb la construcció de piràmides. Tal millora de l'algorisme fast s'ha anomenat oFAST *(oriented* FAST*).*

2.5.1.1. Detector FAST

Per començar l'algorisme realitza la detecció de punt d'interès utilitzant el mètode *FAST,* el qual s'ha explicat anteriorment a 2.2.5.

Com que *FAST* no produeix cap mesura de la resposta d'una cantonada, s'utilitza la funció *Harris corner* (explicada en 2.2.3) per a donar un valor a cadascuna i posteriorment poder-les ordenar en funció d'aquest. Es recorda la funció de *Harris corner*, R:

$$R = det(M) - \alpha\, trace(M)^2 = \lambda_0\lambda_1 + \alpha(\lambda_0 + \lambda_1) \tag{2.5.1}$$

On:

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x & I_xI_y \\ I_yI_x & I_y \end{bmatrix} \tag{2.5.2}$$

Resumint, donat un valor de llindar d'intensitat es produeixen *K* punts d'interès a partir de *FAST*. Aquests s'ordenen en funció de R i finalment es retenen els *N* millors punts com a punts d'interès.

Per altre banda, FAST tampoc produeix punts d'interès a diferents escales. Es proposa utilitzar una piràmide d'escala tal com la de la Figura 2.5.1 i extreure'n punts de cada nivell.

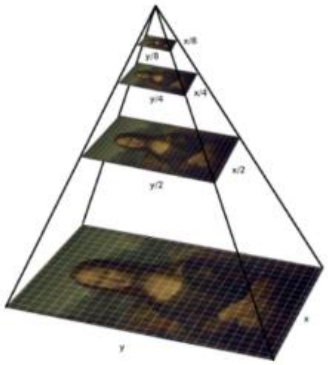


Figura 2.5.1. Piràmide per a extreure punts de diferents escales. A cada nivell la resolució és disminuïda per quatre. (Font: [4])

2.5.1.2. Orientació a partir del centroide d'intensitat

El mètode per a determinar l'orientació assumeix que el centroide d'intensitat queda desplaçat del centre i s'utilitza la localització d'aquest per a calcular l'orientació.

Els moments d'una regió es defineixen com:

$$m_{pq} = \sum_{x,y} x^p \, y^q \, I\,(x,y) \tag{2.5.3}$$

I amb els moments es calcula el centroide, tal que:

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}}\right) \tag{2.5.4}$$

Definint el vector OC amb origen O al centre de la regió (corresponent al punt d'interès) i fi al centroide C, l'orientació de la regió és simplement:

$$\theta = \; atan2\,(m_{01}\,, m_{10}) \tag{2.5.5}$$

On $atan2$ és l'anomenada arctangent de dos paràmetres, caracteritzada per donar el valor de l'arctangent però sense perdre el signe en la divisió.

Aquest càlcul es fa en una regió circular al voltant del punt de radi *r*.

2.5.2. Descriptor rBRIEF

El descriptor que utilitza *ORB* és anomenat rBRIEF *(rotation-aware* BRIEF*)*. Com el nom indica, està basat en el descriptor BRIEF, explicat a l'apartat 2.2.6, però tenint en compte la rotació. També adopta un conjunt de localitzacions de test diferent.

Recordem que el descriptor BRIEF és construït a partir d'un conjunt de tests d'intensitat entre parells de píxels dins d'una regió al voltant dels punts detectats prèviament, formant un vector de bits. El test τ sobre la regió *p* es defineix com:

$$\tau(p;x,y) := \begin{cases} 1, & si\ I(x) < I(y) \\ x, & Altrament \end{cases} \tag{2.5.6}$$

I el descriptor, amb un conjunt de n_d (x,y) parells de localitzacions de test:

$$f_{n_d}(p) = \sum_{1 \le i \le n_d} 2^{i-1}\ \tau(p;x_i,y_i) \tag{2.5.7}$$

Per a aconseguir una invariància a la rotació, es proposa el mètode anomenat *steer BRIEF*, el qual té en consideració l'orientació dels punts. Per a un conjunt de *n* tests a les localitzacions (x_i, y_i), es defineix la matriu S tal que:

$$S = \begin{pmatrix} x_1 & \dots & x_n \\ y_1 & \dots & y_n \end{pmatrix} \tag{2.5.8}$$

Seguidament, utilitzant l'orientació de la regió θ al voltant del punt d'interès i la matriu de rotació R_θ, es construeix la nova matriu de localitzacions de test S_θ

$$S_\theta = R_\theta\, S \tag{2.5.9}$$

Finalment, el descriptor *steer* BRIEF es defineix com:

$$g_n(\,p,\theta) :=\ f_n(p) | (x_i,y_i) \in\ S_\theta \tag{2.5.10}$$

2.5.2.1. Aprenentatge del conjunt de localitzacions de test

Interessa que existeixi una gran variància entre els bits dels descriptors, d'aquesta manera hi ha més discriminació entre les descripcions dels diferents punts. Un altre propietat desitjada és que aquests tinguin una correlació mínima, així cada test contribueix diferentment al resultat.

Una important característica del descriptor BRIEF és que cada bit del descriptor té una gran variància, traduït en una mitjana propera a una meitat. Per altre banda, els bits obtinguts amb el mètode *steer BRIEF* tenen una variància menor i la mitjana distribuïda més uniformement, com s'observa a la Figura 2.5.2. Per a solucionar aquest problema, es va desenvolupar un mètode d'aprenentatge automàtic per a triar un conjunt de tests binaris que maximitzin la variància i minimitzin la correlació dels bits produïts pels tests binaris. *L'steered BRIEF* amb aquest conjunt de tests és anomenat *rotated BRIEF* i és el que l'ORB utilitza. En el següent histograma s'observen les mitjanes dels diferents mètodes.

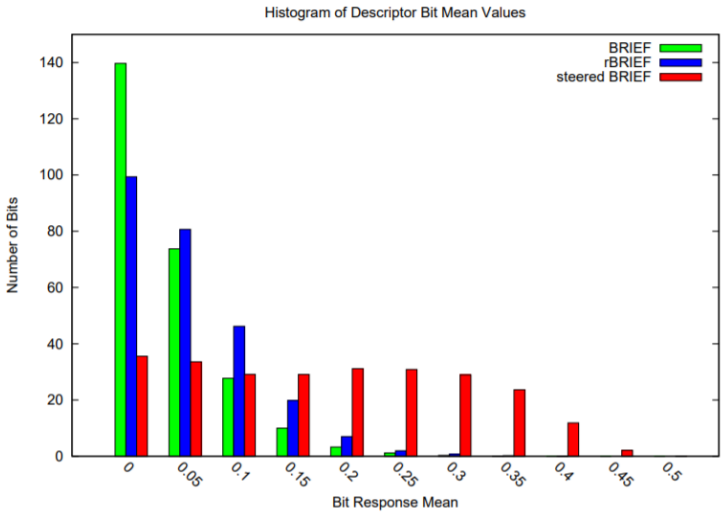


Figura 2.5.2. Histograma de la mitjana dels bits dels diferents descriptors. L'eix *x* representa la distancia a la mitjana de 0,5. (Font: [4])

El mètode d'aprenentatge es va executar sobre 300 mil punts de diferents imatges i s'explica a continuació.

Primerament, s'anomenen tots els possibles tests binaris dins una regió de 31x31 píxels. Com anteriorment, cada test és un parell de 5x5 sub-regions de la regió principal. D'aquesta manera es tenen *M* = 205590 possibles tests. L'algorisme és el següent:

1. Executar tots els possibles tests sobre cada regió dels 300 mil punts d'entrenament i guardar el resultats al vector *T.*
2. Ordenar *T* en funció de la distancia a una mitjana de una meitat.
3. Cerca d'òptims:
    a. Posar el primer test al vector de resultats R i esborrar-lo de T.
    b. Comparar el pròxim test de *T* amb tots els tests de *R*. Si la correlació és major d'un cert llindar, es descarta. Sinó s'afegeix a *R*.
    c. Repetir els passos anteriors fins que es tinguin 256 tests a *R.* Si n'hi ha menys de 256, s'augmenta el llindar de correlació i es prova de nou.

2.6. Detectors i descriptors a OpenCV

Per a utilitzar els algorismes de detecció, descripció i emparellament de la llibreria *OpenCV*, primerament cal veure i entendre com aquests estan estructurats dins d'ella.

S'observa que existeix una interfície comuna per a tots els detectors, una altre per a tots els descriptors i una última per a les tècniques d'emparellament. Aquesta interfície permet a l'usuari canviar fàcil i ràpidament entre diferents algorismes que resolguin la mateixa tasca.

Tots els objectes de detectors i descriptors es generen a partir de les classes abstractes anomenades *Feature Detector* i *DescriptorExtractor*. Aquestes contenen els mètodes *detect*, *compute* i *detectAndcompute*, amb els quals es farà la crida als algorismes per detectar i descriure (separadament o a la vegada).

Per a poder cridar els mètodes, primerament s'ha d'haver creat l'objecte, heretat de les classes que s'acaben d'esmentar. Per a fer-ho, es disposa de diferents creadors d'objectes per als diferents algorismes que conté la llibreria. Aquests permeten crear-los en funció de diferents paràmetres.

Resumint, per a realitzar tasques de detecció i descripció utilitzant la llibreria *OpenCV* només cal crear l'objecte amb el creador de l'algorisme que es vulgui utilitzar (en aquest treball s'utilitzaran *SIFT*, *SURF* i *ORB)* i posteriorment executar el mètode dels esmentats anteriorment per a realitzar les deteccions o descripcions. A continuació, s'explicarà com es creen els creadors d'objectes de cadascun.

2.6.1. SIFT

Com s'ha explicat a l'apartat 2.6, el primer pas per a realitzar tasques de detecció i descripció és crear l'objecte del algorisme que es desitgi, especificant els paràmetres d'aquest. El constructor d'objectes SIFT és el següent:

```
sift=cv.xfeatures2d.SIFT_create(  nfeatures=0,
                                  nOctaveLayers=4,
                                  contrastTreshold=0.03,
                                   edgeTreshold=10,
                                  sigma=1.6)
```

Els diferents paràmetres són:

- *nfeatures*: El nombre de punts a retenir. Escull els *n* millors punts avaluant el contrast amb l'equació (2.3.7). Si s'especifica un nombre no s'utilitza el paràmetre *contrastTreshold*. Per defecte és zero, significant que es retornen els punts que compleixin els llindars *contrastTreshold* i edge*Treshold*.
- *nOctaveLayers:* Nombre d'escales per a cada octava de diferències de gaussianes.
- *contrastTreshold:* Valor del llindar de contrast, comparat amb el valor avaluat de l'equació (2.3.7). Utilitzat per rebutjar punts amb poc contrast.
- *edgeTreshold:* Valor del llindar per a eliminar punts amb una curvatura major a l'altre, explicat a l'apartat 2.3.1.3. És comparat amb l'expressió (2.3.11).
- *sigma:* Valor de la desviació tipus, σ, de la gaussiana aplicada a la imatge de la primera octava.

Els paràmetres recomanats a l'article [1] són els que venen per defecte, així que no se'n modificarà cap alhora de crear l'objecte SIFT en el posterior estudi.

2.6.2. SURF

Per aquest algorisme, el creador és el següent:

```
SURF=cv.features2D.SURF.create(     double hessianTreshold = 100,
                                    int nOctaves = 4,
                                    int nOctaveLayers=4,
                                    bool extended = false,
                                    bool upright=false )
```

On els diferents paràmetres són:

- hessianTreshold: Llindar del detector hessià. Els punts amb valor del determinant de la matriu Hessiana avaluada al punt i escala corresponent amb valor menor a aquest llindar seran rebutjats. Explicat amb més detall a l'apartat 2.4.1.
- nOctaves: Nombre d'octaves de la piràmide construïda a partir de la imatge d'entrada.
- nOctaveLayers: Nombre d'escales dins de cada octava.
- extended: Si el boleà és *true* s'utilitzen els descriptors ampliats (de 128 dimensions), si és *false* s'utilitzen els descriptors estàndards de 64 dimensions.
- Upright: Si és *true* es calcula l'orientació, sinó no.

Al posterior estudi es deixaran els valors per defecte ja que tots estan dins del rang recomanat a l'article [2]. Es farà però l'anàlisi del canvi de la versió extensa de 128 dimensions en comparació a l'estàndard de 64.

2.6.3. ORB

El constructor del detector i descriptor *ORB* és el següent:

```
Fast = cv.ORB_create(   int nfeatures=500,
                        float scaleFactor = 1.2,
                        int nlevels = 8,
                        int firstLevel=0,
                        int WTA_K=2
                        int scoreType=HARRIS_SCORE
                        int patchSize=31,
                        int fast Treshold=20 )
```

Els paràmetres fan referència a:

- *nfeatures*: El màxim nombre de punts a retenir. A l'apartat 2.5.1.1 s'explica el criteri establert al realitzar aquesta retenció.
- *scaleFactor*: Factor d'escalat de la piràmide. Per exemple, si el factor és dos, el pròxim nivell de la piràmide tindrà quatre vegades menys de píxels.
- *nlevels*: Nombre de nivells de la piràmide.
- *firstLevel*: El nivell de la piràmide a on es posa la imatge d'entrada. Les escales inferiors seran imatges amb major resolució.
- *WTA_K*: El valor per defecte de dos indica que a cada test es comparen dos píxels, tal i com s'explica a l'article [ORB]. Es poden comparar dos, tres o quatre píxels per test, generant descriptors de 2 bits per test.
- *scoreType*: Per defecte és *HARRIS_SCORE*, indicant a l'algorisme que avaluí els punts amb la funció *Harris corner* introduïda (2.2.7), per a posteriorment retenir els *n* millors. Una alternativa és utilitzar *FAST_SCORE* que utilitza el mètode explicat a 2.2.5 per a donar una puntuació als punts i poder-los ordenar en funció d'aquesta.
- *patchSize*: Mida de la regió, en píxels, que utilitza l'*oriented BRIEF* al primer nivell de la piràmide.
- *fastTreshold*: Llindar utilitzat en les comparacions d'intensitats de píxels en els tests que avalua el detector FAST. De nou, per més detalls del llindar consultar l'apartat 2.2.5.

De nou, la llibreria estableix per defecte uns valors adequats. En el posterior estudi l'únic paràmetre que es modificarà és el nombre de punts a retenir, que s'adaptarà en funció dels valors que proporcionin SIFT i SURF i la resolució de la imatge.

2.7. Feature Matching

Una vegada s'han extret els punts característics i els descriptors de dos o més imatges, el pròxim pas és establir emparellaments entre els punts d'aquestes.

Cal introduir les dos distàncies que s'utilitzaran en els algorismes d'emparellament, aquestes són la distància euclidiana i *Hamming*. Donat dos punts *n*-dimensionals $P = (p_1, p_2, \dots p_n)$ i $Q = (q_1, q_2, \dots q_n)$, la distància euclidiana és la distancia "ordinària", deduïda a partir del teorema de Pitàgores, definida com:

$$d_{euclidiana}(P, Q) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2} \tag{2.7.1}$$

La distància *Hamming* entre dues cadenes de la mateixa longitud fa referència al nombre de posicions on hi ha un valor diferent. Al considerar cadenes de bit, es correspon al nombre de bits que s'han de canviar d'una cadena perquè adopti el valor de l'altre en relació al total. Així doncs, donats dos vectors o cadenes de bits P i Q es defineix la distància *Hamming* de la següent manera:

$$d_{Hamming}(P, Q) = \frac{\sum_{i=0}^{n} \delta(p_i, q_i)}{n} \tag{2.7.2}$$

On:

$$\delta(p_i, q_i) = \begin{cases} 1, & si\ p_i = q_i \\ 0, & si\ p_i \neq q_i \end{cases} \tag{2.7.3}$$

Aquesta és molt ràpida de calcular ja que per cadenes de bits s'obté directament amb la suma dels bits resultants de l'operació *XOR* entre les cadenes, per aquest motiu es sol utilitzar per a descriptors binaris com el *BRIEF* i l'*ORB*, mentre que per *SIFT* i *SURF* s'utilitza la euclidiana.

Una vegada definides les distàncies, s'explica en què consisteix l'emparellament. Primerament, s'assumeix que els descriptors has estat dissenyats per tal que la distancia euclidiana o *Hamming* d'aquests pugui ser utilitzada directament per a trobar emparellaments potencials. Donada una distancia, l'estratègia més fàcil podria ser fixar una distancia llindar i retornar tots els emparellaments de l'altre imatge amb distancia menor al llindar fixat. Com és lògic, si es fixa el llindar massa alt s'aconsegueixen masses falsos positius, i fixant-lo massa baix resulta en masses falsos negatius.

Si es disposes de suficient dades d'entrenament, es possible aprendre i adaptar aquest llindar per a diferents punts. Normalment, i com és el cas d'aquest estudi, només es disposa d'una col·lecció

d'imatges. En aquest cas, una altre estratègia seria simplement emparellar-lo amb el veí més pròxim. Com que podria ser que molts punts no tinguessin cap emparellament real a l'altre imatge, també s'hi hauria d'afegir un llindar per reduir el nombre de falsos positius.

Un altre i efectiu mètode es comparar la distancia del veí més pròxim amb la del segon. Aquest mètode funciona bé ja els emparellaments correctes necessiten tenir el veí més proper significativament més a prop que l'emparellament incorrecte més proper. Per a els falsos emparellaments, és molt probable que n'existeixin altres de falsos amb una distància similar degut a les altes dimensionalitats dels descriptors.

En la següent imatge es pot veure que fixant un llindar el descriptor D_A falla al emparellar D_B i el descriptor D_D falla amb D_C i D_E. Agafant el veí més proper D_A emparella correctament D_B però D_D ho incorrectament al agafar D_C. Finalment, si es compara la distancia del primer veí amb la del segon, D_A emparella correctament D_B, ja que d_1 és molt major a d_2 i D_D rebutja correctament D_C i D_E perquè d_1 és pròxima a d_2.

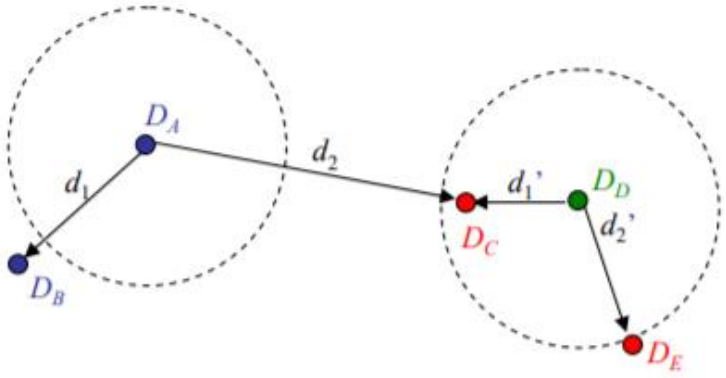


Figura 2.7.1. Diferents mètodes per establir emparellaments. (Font: [6]).

Una altre opció és l'anomenada *crosscheck*, la qual només es donen per bons els emparellaments (i, j) els quals el descriptor i de la primera imatge correspongui al descriptor j de la segona i viceversa. És dir, els dos punts de les dos imatges han d'emparellar-se entre ells.

2.8. Feature Matching a OpenCV

Open CV, com s'ha dit anteriorment, té una interfície comuna per a l'emparellament de descriptors que et permet canviar fàcilment entre diferents algorismes. Al fer l'emparellament, es retorna un objecte amb estructura anomenada *DMatch*, la qual té les següents propietats d'interès:

- *DMatch.distance*: Distancia entre els descriptors emparellats.
- *DMatch.trainIdx*: Índex del descriptor de la primera imatge.
- *DMatch.queryIdx* : Índex del descriptor de la segona imatge.

Per a realitzar l'emparellament entre dos descriptors, primerament s'ha de crear l'objecte de la classe de emparellament que es desitgi, i posteriorment es fa la crida a l'algorisme que es vulgui aplicar (veí més pròxim, *k* veïns més pròxims, etc). A continuació s'explicaran les dos classes que hi ha disponibles a la llibreria *Open CV.*

2.8.1. Brute Force Matcher

Per a cada punt del primer descriptor, troba els *k* punts més pròxims del segon descriptor calculant les distàncies entre tots els punts. Al calcular totes les distàncies, pot resultar molt lent per a descriptors de gran longitud.

Com s'ha dit anteriorment, primerament es necessari crear l'objecte de la classe *BFMatcher*. Al crear-lo es pot especificar el tipus de distancia a utilitzar i si es desitja que realitzi *crosscheck*. El constructor d'aquest es veu a continuació:

```
BFmatcher=cv.BFMatcher(     int normType=NORM_L2,

                            bool crossCheck=false)
```

Els paràmetres fan referència al següent:

- *normType*: Es defineix el tipus de distància a mesurar. Són s'utilitzaran *NORM_L2* i *NORM_HAMMING*, les quals corresponen a la distància euclidiana i *Hamming.*
- *crossCheck*: Si aquest paràmetre és assignat verdader es retornaran els emparellaments utilitzant la tècnica de *crosscheck*. Si és fals, es retornaran considerant solament les distàncies.

2.8.2. Fast Approximate Nearest Neighbor Search

Aquesta classe conté una col·lecció d'algorismes optimitzats per a trobar veïns pròxims més ràpidament en grans conjunts de dades i de gran dimensionalitat. Cal destacar que amb aquest mètode es troben els veïns més pròxims amb alta probabilitat, mentre que el mètode vist anteriorment assegurava retornar el veí més pròxim al calcular totes les distàncies.

Al crear l'objecte s'han d'especificar dos diccionaris amb els quals s'especifica l'algorisme a utilitzar i certs paràmetres. No s'entrarà en detall en els algorismes i paràmetres que utilitza *FLANN*, sinó que s'utilitzaran els proposats a la documentació de la llibreria OpenCV els quals proporcionen bons resultats.

2.8.3. Algorismes d'emparellament utilitzats en el posterior estudi

En l'estudi que es farà posteriorment, s'utilitzaran els tres mètodes següents, els quals es defineixen i se'n farà referència com:

- Emparellament BF: S'utilitzarà l'algorisme *Brute Force* per a trobar els 2 veïns amb distàncies menors. Si la distància del veí més proper és menor al 70% de la distància del segon, s'efectuarà l'emparellament. Aquest mètode es proposa a [1].
- Emparellament BF *crosscheck*: S'utilitzarà l'algorisme *Brute Force* amb *crosscheck*.
- Emparellament *FLANN*: S'utilitzarà l'algorisme FLANN per a trobar els 2 veïns amb distàncies menors. Una altre vegada, si la distància del veí més proper és menor al 80% de la distància del segon, s'efectuarà l'emparellament.

3. Comparativa dels diferents mètodes

Per a realitzar una comparació dels diferents algorismes explicats anteriorment (en detall, es realitzarà la comparació de SIFT, SURF, la versió extensa de SURF i ORB) s'ha creat un conjunt d'imatges amb diferents transformacions. El conjunt d'imatges, posteriorment anomenat *data set,* està format per 7 sub-conjunts, cadascun corresponent a una transformació en particular.

Cada sub-conjunt té una imatge inicial o patró, sobre la qual s'hi aplica la transformació corresponent incrementada gradualment 50 vegades, obtenint així 50 imatges per a cada sub-conjunt. Aquest procés s'explica detalladament a l'apartat 3.1.

Seguidament, s'aplicaran els diferents algorismes de detecció descripció i emparellament entre els 350 parells d'imatges (imatge patró amb les 50 transformacions, dels 7 sub-conjunts) i s'extrauran i calcularan diferents propietats, explicades a 3.2.1, per a analitzar i avaluar el funcionament dels algorismes sota les transformacions pertinents. Aquest anàlisi es realitza a l'apartat 3.3.

Cal remarcar que les dades s'han obtingut amb un ordinador amb les següents prestacions:

Tipus sistema	Processador	RAM	Sistema operatiu
64 bits	Intel Core i5-8250U CPU 1,80 GHz	8 GB	Windows 10

Taula 2. Resum dels resultats dels diferents tests pels diferents objectes.

3.1. Creació del *dataset* d'imatges per a la comparació.

En aquest apartat s'explicarà quin és i com s'ha obtingut el data set d'imatges. Cal remarcar que hi ha molts *datasets* disponibles a internet, però aquests no proporcionen les matrius de transformació entre les diferents imatges, cosa que impossibilita l'obtenció de propietats d'interès de forma precisa.

En els següent apartats s'explicaran les diferents transformacions aplicades i es mostraran algunes de les imatges obtingudes, per a més detall a l'annex A2 es mostren totes les imatges del conjunt.

3.1.1. Suavitzat gaussià

Per a aconseguir un efecte de difuminat en les imatges s'ha utilitzat un filtre gaussià amb diferents valors de desviació tipus per anar augmentant l'efecte.

Es recorda, que la funció gaussiana de dos dimensions és la següent:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{3.1.1}$$

És necessari discretitzar-la per a aconseguir un *kernel* i poder filtrar la imatge amb la convolució d'aquest. Teòricament, es necessitaria un *kernel* infinit, ja que la distribució gaussiana pren valors majors que zero en tot l'espai. Afortunadament, casi bé el 99% de la distribució queda dins de la regió amb valors inferiors a tres desviacions estàndard. Per tant, es sol limitar la mida del *kernel* per tal que només contingui els valors d'aquesta regió. Finalment, es divideix el *kernel* per el valor que provoqui una mitjana unitària. A continuació es pot veure el *kernel* gaussià amb desviació estàndard unitària.

$$\frac{1}{273}\begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix}$$

Figura 3.1.1. *Kernel* gaussià amb desviació unitària. (Font: Elaboració pròpia).

Per a filtrar imatges amb gaussianes, *Open CV* proporciona la funció *Gaussian Blur*. S'han obtingut 50 imatges amb gaussianes amb desviacions entre 0 i 5. A continuació es mostren algunes de les imatges obtingudes:

Figura 3.1.2. D'esquerra a dreta: Imatge inicial, imatge amb $\boldsymbol{\sigma}$=2,5 i imatge amb imatge amb $\boldsymbol{\sigma}$=5. (Font: Elaboració Pròpia).

3.1.2. Rotació

Definint els punts de 2 dimensions (coordenades de píxels a una imatge) com:

$$x = \begin{bmatrix} \mathrm{x} \\ \mathrm{y} \end{bmatrix} \tag{3.1.2}$$

I la matriu de transformació geomètrica de rotació com:

$$R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{3.1.3}$$

La localització dels punts $\hat{x}$ de la imatge rotada θ s'obtenen amb la multiplicació dels punts x de la imatge d'entrada amb els de la matriu de rotació $R(\theta)$.

$$\hat{x} = R(\theta)x \tag{3.1.4}$$

S'han obtingut 50 imatges rotades entre 0 i 180 graus. A continuació se'n mostren tres:

Figura 3.1.3. Esquerra: Imatge inicial. Restants: Dos imatges rotades del conjunt elaborat. (Font: Elaboració Pròpia).

3.1.3. Canvi d'escala

El canvi d'escala es realitzarà augmentant o disminuint la resolució de la imatge patró. Donant els píxels x de la imatge d'entrada, i definint la matriu de transformació de canvi d'escala com:

$$E(f) = \begin{bmatrix} f & 0 \\ 0 & f \end{bmatrix} \tag{3.1.5}$$

Els punts $\hat{x}$ de la imatge escalada un factor f s'obtenen amb la multiplicació dels punts x amb la matriu de canvi d'escala.

$$\hat{x} = E(f)x \tag{3.1.6}$$

Si es fa un escalat positiu, hi haurà certs píxels $\hat{x}$ de la imatge escalada als quals no se'ls hi ha assignat cap valor amb la multiplicació. Per contrast, al fer un escalat negatiu hi haurà certs píxels $\hat{x}$ els quals se'ls hi hauria d'assignar dos o més valors. Existeixen diferents tècniques per a abordar aquests problemes. Per a escalats positius es solen utilitzar interpolacions bicúbiques i bilineals, mentre que per a escalats negatius es sol utilitzar el mètode anomenat mitjana d'àrees, el qual fa una mitjana ponderada dels diferents píxels de la imatge inicial que s'assignen als de la imatge escalada.

S'ha obtingut un conjunt de imatges amb escalat positiu amb factors entre 1 i 4, i un darrer amb factors entre 1 i un quart. Amb interpolació bicúbiques per al positiu i amb mitjana d'àrees pel negatiu. A continuació, es mostren algunes de les imatges els dos conjunts.

Figura 3.1.4. Tres imatges del conjunt d'escalat positiu. (Font: Elaboració Pròpia).

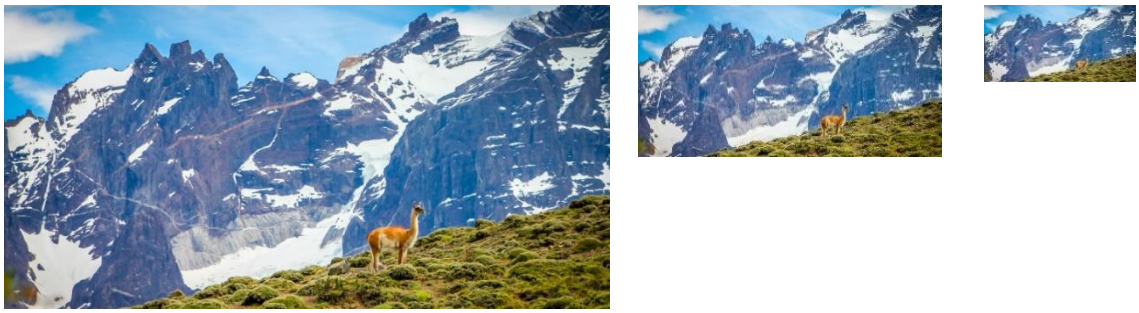


Figura 3.1.5. Tres imatges del conjunt d'escalat negatiu. (Font: Elaboració Pròpia).

3.1.4. Perspectiva

Per a produir l'efecte d'un canvi de punt de vista de la imatge, es durà a terme una transformació perspectiva. Per a realitzar aquesta transformació, és necessari treballar amb coordenades homogènies, ja que permeten manipular vectors de dos dimensions (imatges) en un espai de tres.

Un punt bidimensional es converteix a coordenades homogènies i viceversa de la següent manera:

$$\begin{bmatrix} x \\ y \end{bmatrix} \rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \qquad \begin{bmatrix} x \\ y \\ w \end{bmatrix} \rightarrow \begin{bmatrix} \frac{x}{w} \\ \frac{y}{w} \end{bmatrix} \tag{3.1.7}$$

Definint la matriu de transformació perspectiva com:

$$H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \tag{3.1.8}$$

Operant amb coordenades homogènies, els punts $\widetilde{x}'$ de la imatge transformada s'obtenen amb la multiplicació de la matriu de transformació amb els punts $\widetilde{x}$ de la imatge inicial. Es recorda que per aconseguir els punts $\widetilde{x}$ s'ha de realitzar la conversió explicada en (3.1.7).

$$\widetilde{x}' = H\,\widetilde{x} \tag{3.1.9}$$

Finalment, només cal convertir els punts de la imatge transformada $\widetilde{x}'$ en coordenades bidimensionals, tal i com s'ha esmentat anteriorment.

El procés complert es pot realitzar amb una sola operació, tal que:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}} \\ \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}} \end{bmatrix} \quad (3.1.10)$$

Aquesta operació és la que realitza *OpenCV*, amb la funció *wrapperspective*. Ara bé, s'ha d'escollir una matriu de transformació H que produeixi l'efecte que es desitja.

Per a escollir-la ha set d'utilitat la funció *getprespectiveTransfom*. Donats quatre punts de la imatge inicial i quatre de la imatge transformada, aquesta funció troba una matriu de transformació H que projecti els punts de la imatge inicial a els punts de la transformada.

Per a una comprensió visual, s'adjunta la següent imatge. Seleccionant els quatre punts de la imatge inicial i quatre més de la segona imatge, la funció *getprespectiveTransfom* calcula la matriu de transformació H. Finalment, es realitza la operació (3.1.10) amb la funció *wrapperspective*, obtenint la imatge de la dreta.

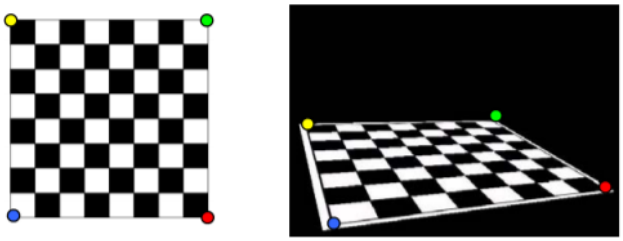


Figura 3.1.6. Exemple de transformació perspectiva. (Font: Elaboració Pròpia).

Així doncs, per a generar el conjunt d'imatges del *data set*, s'han seleccionat els quatre extrems de la imatge inicial i els de la imatge transformada s'han anat modificant gradualment per a obtenir perspectives cada cop més accentuades. A continuació, les imatges resultants:

Figura 3.1.7. D'esquerra a dreta: Imatge inicial, imatge 25 del conjunt de transformació perspectiva i imatge amb la transformació més accentuada del conjunt. (Font: Elaboració Pròpia).

3.1.5. Lluminositat

Per a explicar com s'ha modificat lluminositat de la imatge inicial, primerament cal introduir el model de color HSV.

El model de color HSV (*Hue, Saturation, Value,* en català: Matriu, Saturació, Valor) , també anomenat HSB (*Hue, Saturation, Brightness,* en català: Matriu, Saturació, Lluminositat), és un model que defineix els colors amb aquestes tres components. Seguidament, es veu la representació de l'espai de colors d'aquest model, en coordenades cilíndriques:

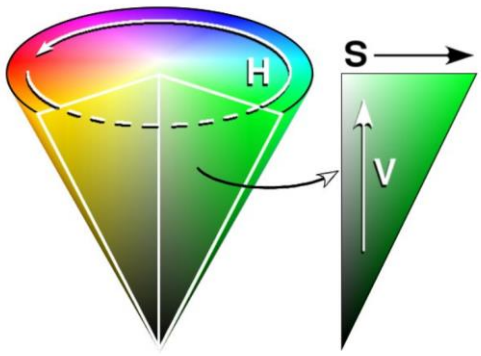


Figura 3.1.8. Representació de l'espai de colors HSV. (Font: [6]).

Les diferents variables expressen:

- Matriu (*H*): Es representa com un angle amb rang entre 0 i 360 en graus. Cada valor correspon a un color. Per exemple: 0º equival al vermell, 120º al verd i 240º al blau.
- Saturació (*S*): Es representa com a la distancia a l'eix vertical blanc-negre. El rang de valors és de 0 a 100%. Com menor sigui el la saturació d'un color, major tonalitat grisosa. Per contrast, com major sigui el seu valor, més pur serà el color.
- Lluminositat (*V* o *B*): Representa l'altura de l'eix blanc-negre. El rang és de nou de 0 a 100% i a mesura que es va augmentat, es va augmentant la lluminositat del color.

Així doncs, per a augmentar la lluminositat per a la construcció del *data set* d'imatges, primerament s'ha realitzat la conversió de l'espai *RGB* al *HSV* per posteriorment augmentar la component de lluminositat *V*.

S'ha realitzat un conjunt per a augments de lluminositat i un altre per a decrements. Les imatge resultants es mostren a continuació.

Figura 3.1.9. Dreta: Imatge inicial. Restants: Dos imatges del conjunt amb augments de lluminositat. (Font: Elaboració Pròpia).

Figura 3.1.10. Dreta: Imatge inicial. Restants: Dos imatges del conjunt amb disminucions de lluminositat. (Font: Elaboració Pròpia).

3.2. Obtenció dels resultats experimentals

3.2.1. Mètodes d'avaluació

A continuació s'explicaran les diferents propietats o característiques que s'han utilitzat per a avaluar cada mètode enfront a diferents distorsions i transformacions per tal de poder-ne fer una comparació.

Per a avaluar els detectors, s'han calculat el nombre de punts d'interès, el nombre de correspondències i la repetibilitat entre parells d'imatges. Les correspondències són els punts que s'han detectat com a punt d'interès tant en la primera imatge com en la segona. La repetibilitat es defineix com el nombre de correspondències entre el nombre mínim de punts d'interès de les dos imatges.

Una altre propietat interessant a calcular és l'anomenat *recall*, definit a (3.2.1). Aquest paràmetre és molt interessant, ja que indica el percentatge de correspondències que s'han emparellat. També es calcularà l'exactitud de l'emparellament dels descriptors, obtinguda amb (3.2.2).

$$recall = \frac{TP}{nombre\ correspondencies} \tag{3.2.1}$$

$$exactitud = \frac{TP}{TP + FP} \tag{3.2.2}$$

On:

- TP: *true positives*. Nombre d'emparellaments correctes.
- FP: *false positives*. Nombre d'emparellaments proposats però que són incorrectes.

Finalment, es mesuraran els temps de computació dels diferents algorismes de detecció, descripció i emparellament.

3.2.2. Obtenció de propietats

Com s'ha dit anteriorment, l'objectiu és comparar el funcionament dels detectors i descriptors sota diferents condicions o transformacions. Fins ara, s'ha explicat l'obtenció del *data set* de imatges amb les diferents transformacions i en aquest apartat s'explicarà el mètode dissenyat per a l'extracció de dades per a poder realitzar la comparativa.

Per a estructurar l'algorisme d'extracció de dades, s'ha creat una llista de 350 posicions (referents a les 350 imatges creades del *data set*) on cada posició d'aquesta correspon a una altre llista, amb els següents elements d'interès:

1. Nom de la imatge inicial del sub-conjunt: Nom de la imatge base. És dir, sense cap tipus de transformació.
2. Nom de la imatge *n*: Nom de la imatge transformada un grau *n* del subconjunt.
3. Matriu de transformació: Matriu de transformació aplicada per a transformar la primera imatge en la segona. S'hi introdueix una matriu identitat per a canvis de lluminositat i suavitzat gaussià.

Un cop creada la llista en qüestió és fàcil i immediat llegir les dos imatges i la seva matriu de transformació. D'aquesta manera, amb 350 iteracions es podran detectar, descriure i emparellar les 350 imatges d'interès i extreure'n les propietats descrites en l'apartat anterior.

Abans d'explicar el funcionament de l'algorisme d'extracció de dades, és necessari explicar les dos funcions que s'han implementat en aquest per a trobar l'exactitud i les correspondències dels diferents parells d'imatges.

La primera funció extreu l'exactitud de l'emparellament entre dos imatges. Un cop s'ha fet l'emparellament, es tenen les localitzacions dels punts d'interès d'ambdós imatges, es disposa de la matriu de transformació i es té assignat un valor de llindar de distàncies, la funció et retorna el nombre de *true positives, false positives* i l'exactitud. També ha sigut necessari la creació d'una variable booleana que indiqui si la transformació és perspectiva, per, en aquest cas, tenir en compte que es treballa amb coordenades homogènies. Seguidament s'exposa l'algorisme, representat amb un diagrama de flux a la Figura 3.2.1.

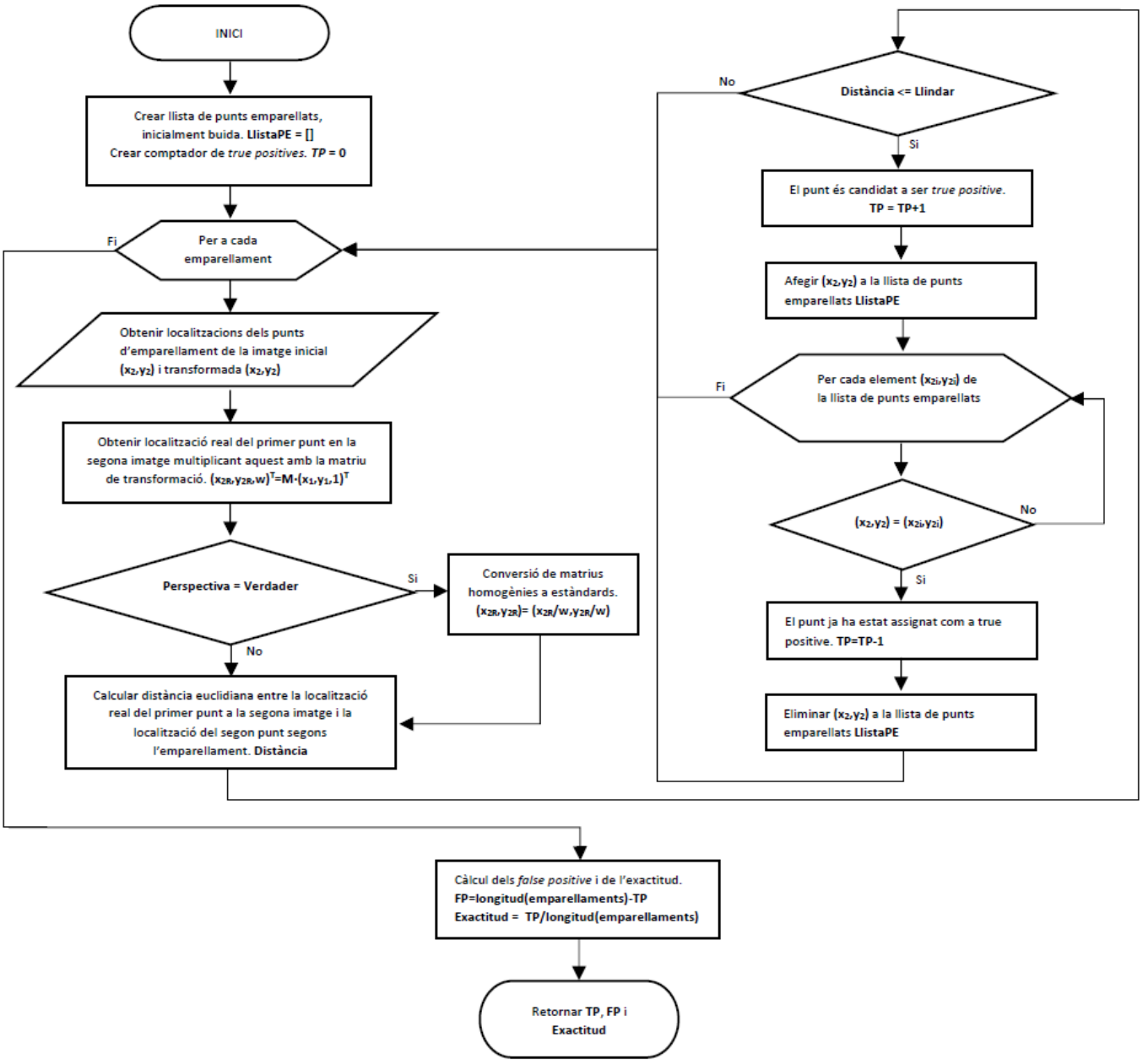


Figura 3.2.1. Diagrama de flux explicant el funcionament de la funció que retorna l'exactitud de l'emparellament. (Font: Elaboració Pròpia).

La segona funció et retorna el nombre de correspondències entre dos imatges. Per a executar-la, es necessiten els punts d'interès de les dos imatges, la matriu de transformació i un llindar de distància. De nou, s'exposa amb un diagrama de flux a la següent figura:

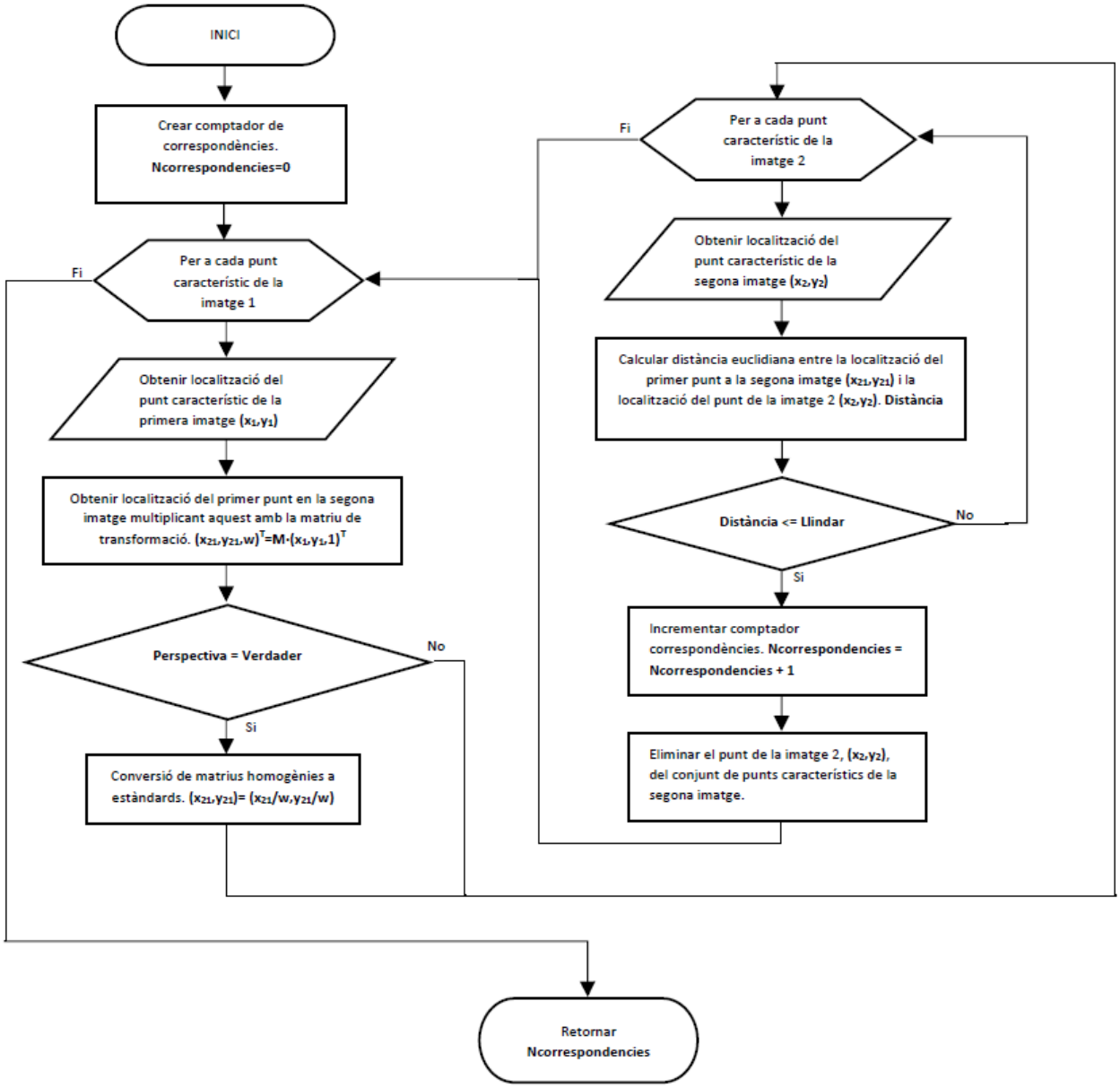


Figura 3.2.2. Diagrama de flux explicant la funció que retorna el nombre de correspondències entre els punts detectats en dos imatges. (Font: Elaboració Pròpia).

Finalment, a la Figura 3.2.3 hi podem veure l'algorisme genèric, el qual exporta totes les propietats desitjades en un fitxer *csv* per a cada algorisme de detecció i descripció. Es recorda que aquests són SIFT, SURF, SURF de 128 dimensions i ORB.

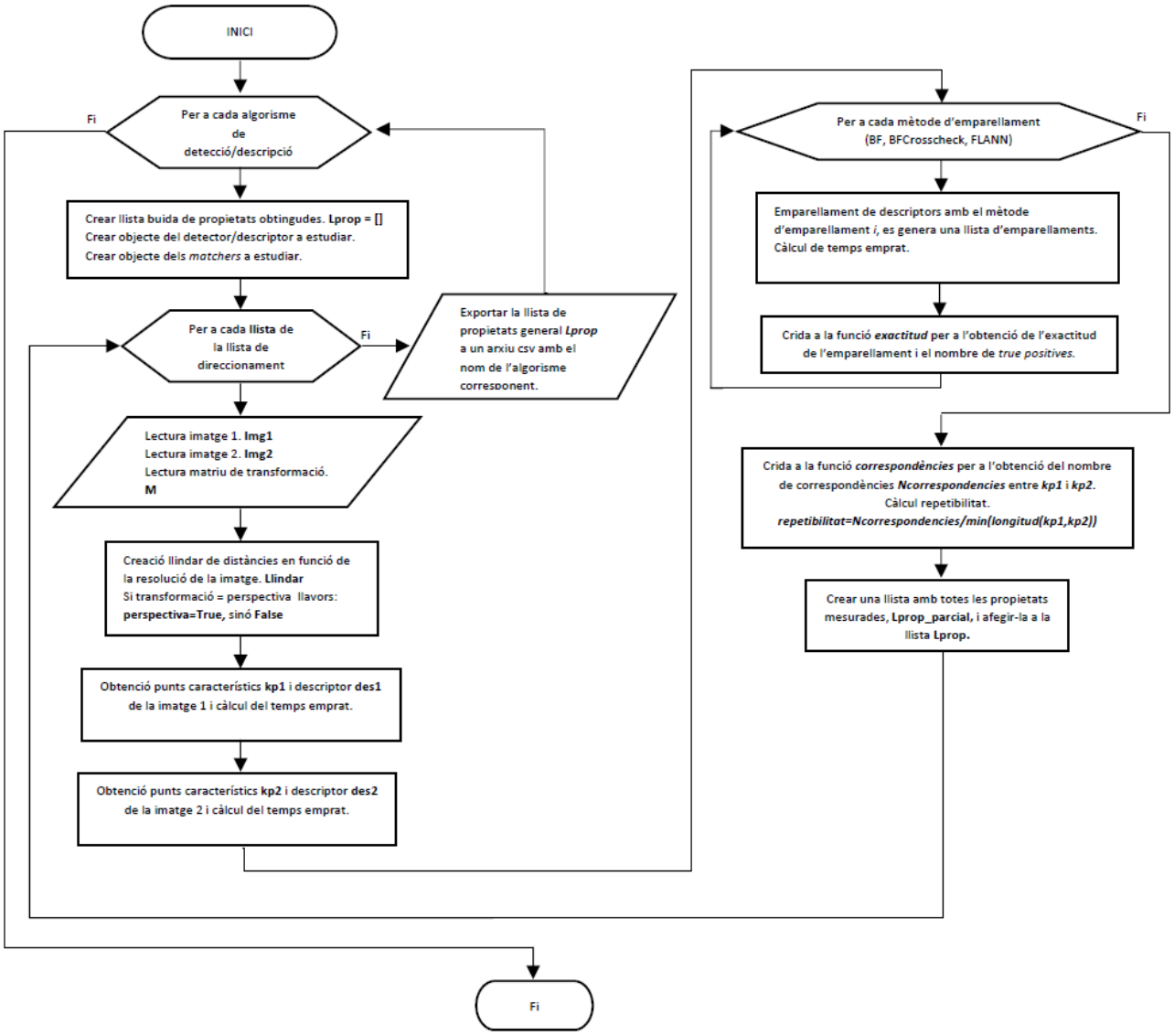


Figura 3.2.3. Diagrama de flux de l'algorisme genèric d'extracció de dades. (Font: Elaboració Pròpia).

En la següent taula es representa part d'un fitxer de propietats exportat, on es veuen totes les propietats obtingudes d'un parell de imatges:

Nom	scaleP12
Temps detecció i descripció imatge 1 (ms)	178.5422
Nombre de punts característics 1	7193
Temps detecció i descripció imatge 2 (ms)	410.0209
Nombre de punts característics 2	17094
Nombre de correspondències	5908
Percentatge de correspondències	0.821354
Temps emparellament BF (ms)	2772.28
Nombre emparellaments BF	5368
***True positives* BF**	4695
***False positives* BF**	673
Temps emparellament BF Crosscheck (ms)	2659.613
Nombre emparellaments BF Crosscheck	6580
***True positives* BF Crosscheck**	4786
***False positives* BF Crosscheck**	1794
Temps emparellament FLANN (ms)	427.9986
Nombre emparellaments FLANN	5366
***True positives* FLANN**	4692
***False positives* FLANN**	674
Exactitud BF	0.874627
Exactitud BF Crosscheck	0.727356
Exactitud FLANN	0.874394
Recall BF	0.794685
Recall BF Crosscheck	0.810088
Recall FLANN	0.794177

Taula 3. Part del fitxer *csv* exportat pel mètode d'extracció de dades dissenyat. Aquest conté totes les propietats obtingudes. (Font: Elaboració pròpia)

3.3. Anàlisi i comparativa dels resultats experimentals

3.3.1. Mètode d'emparellament

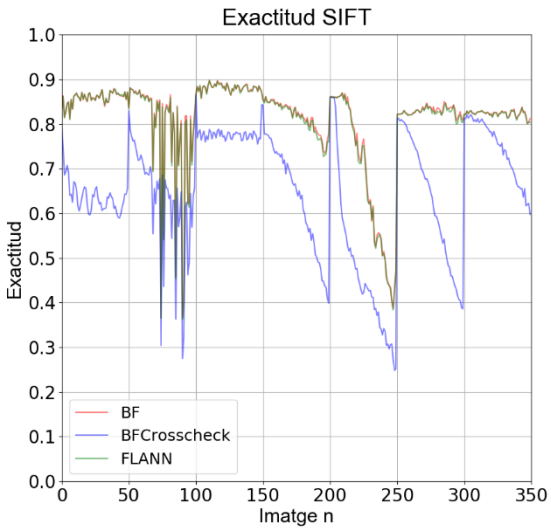
Primerament s'analitzaran els tres mètodes d'emparellament estudiats. Aquests han set introduïts a l'apartat 2.8.3, es recorden quins són i quines són les seves nomenclatures:

- BF: S'utilitzarà l'algorisme *Brute Force* per a trobar els 2 veïns amb distàncies menors. Si la distància del veí més proper és menor al 70% de la distància del segon, s'efectuarà l'emparellament.
- BF *crosscheck*: S'utilitzarà l'algorisme *Brute Force* amb *crosscheck*.
- FLANN: S'utilitzarà l'algorisme FLANN per a trobar els 2 veïns amb distàncies menors. Una altre vegada, si la distància del veí més proper és menor al 70% de la distància del segon, s'efectuarà l'emparellament.

Per a avaluar-los, es representaran les exactituds i temps de cadascun, per a cada algorisme de detecció i descripció.

Com es veu a la Figura 3.3.1, pels descriptors SIFT l'exactitud obtinguda amb BF i FLANN és pràcticament igual i molt superior que l'obtinguda amb BF *crosscheck*.

Pel que fa a la rapidesa de cadascun, s'observa que FLANN és el mètode més ràpid (sobretot quan el nombre d'emparellaments és major). La diferència de temps entre el mètode de retenció en funció de la distància entre els 2 veïns més pròxims i el *crosscheck* és pràcticament nul·la.



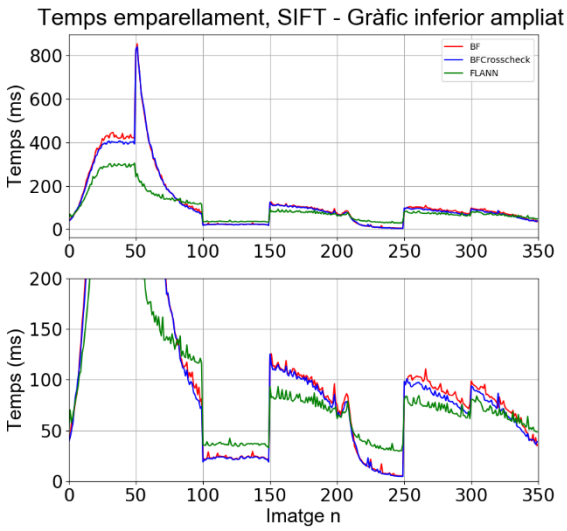


Figura 3.3.1. Exactitud i temps dels diferents mètodes d'emparellament. Descriptors SIFT (Font: Elaboració Pròpia).

Tant en exactitud com en temps els diferents mètodes d'emparellament responen aproximadament d'igual manera amb els descriptors SURF, tant en la versió estàndard de 64 dimensions, com en l'extensa de 128.

S'observa però, que els temps d'emparellament són majors que en SIFT. Aquest augment és degut a que es troben més emparellaments, ja que les dimensions dels descriptors SIFT són de 128 dimensions. Tal augment dels descriptors, fa que FLANN sigui més ràpid gairebé per tot el conjunt d'imatges. També s'aprecia que la versió extensa de SURF triga aproximadament el doble a l'emparellar els descriptors amb els mètodes que utilitzen l'algorisme *Brute force*, ja que s'han de fer el doble de comparacions.

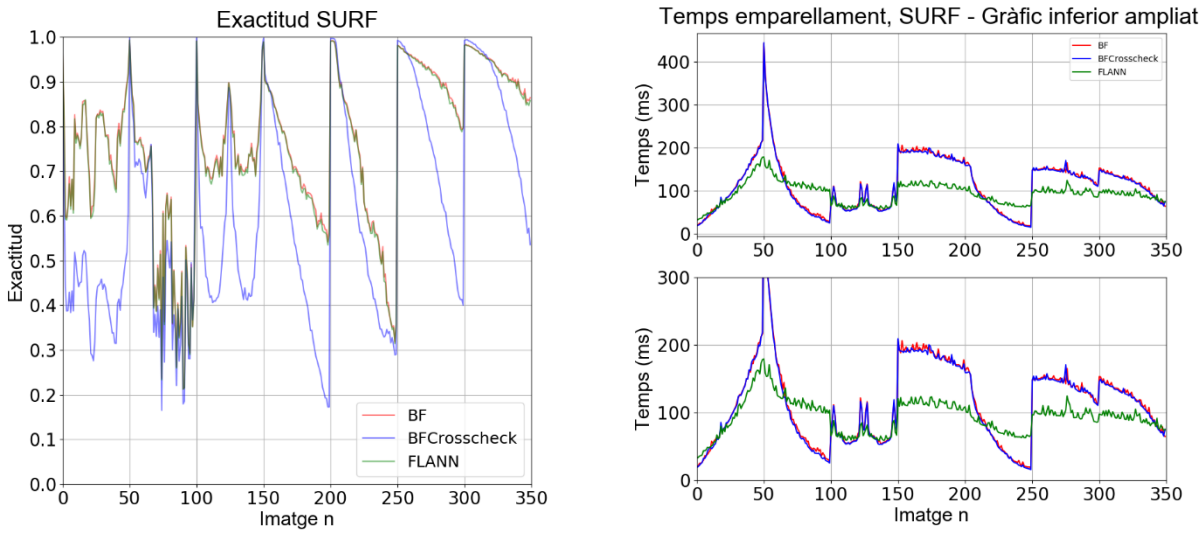


Figura 3.3.2. Exactitud i temps dels diferents mètodes d'emparellament. Descriptors SURF de 64 dimensions. (Font: Elaboració Pròpia).

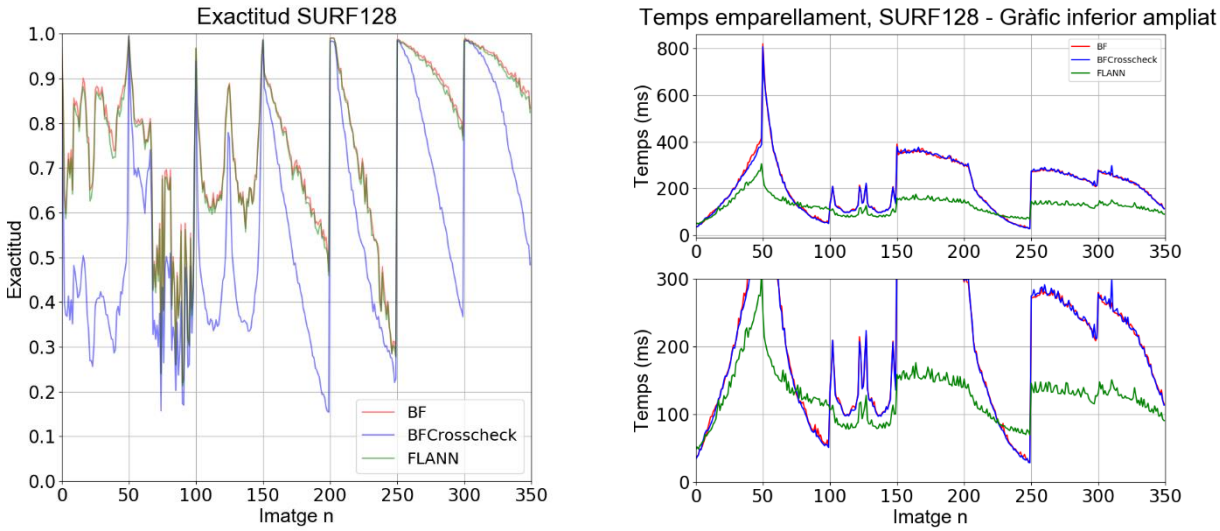


Figura 3.3.3. Exactitud i temps dels diferents mètodes d'emparellament. Descriptors SURF de 128 dimensions. (Font: Elaboració Pròpia).

Els descriptors binaris obtinguts amb ORB també presenten una exactitud semblant que amb els mètodes *BF i FLANN*, i molt superior a la del mètode BF *crosscheck*.

Al ser binaris són els descriptors més ràpids d'emparellar, tot i que, com es veurà més endavant, ORB és el mètode que detecta més punts d'interès. El mètode FLANN, és una altre vegada és el més ràpid.

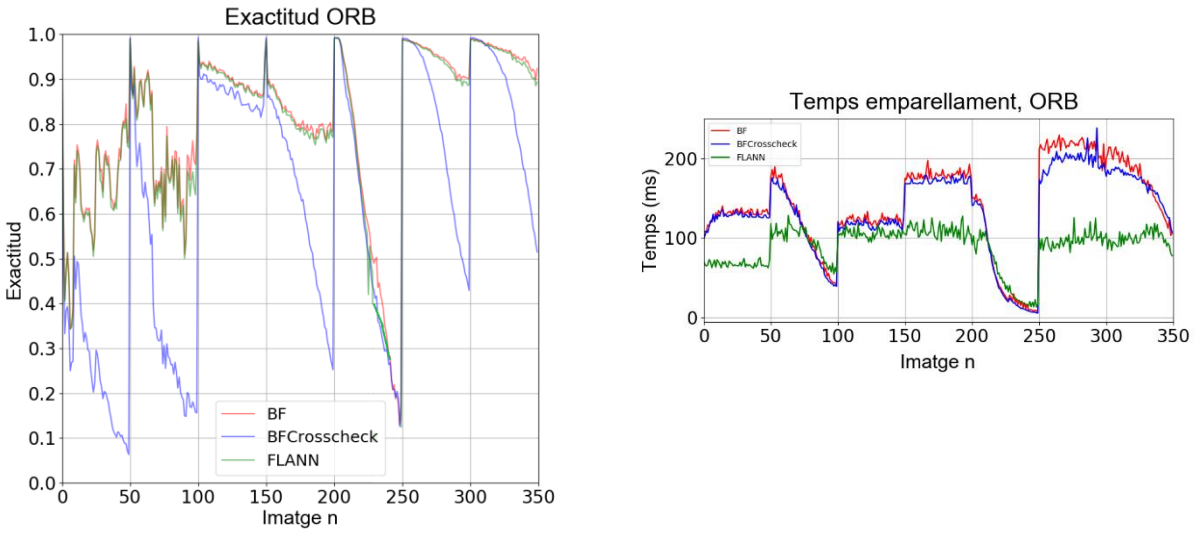


Figura 3.3.4. Exactitud i temps dels diferents mètodes d'emparellament. Descriptors ORB (Font: Elaboració Pròpia).

Tenint en compte els resultats vists amb els 4 descriptors vists anteriorment, es conclou que el millor mètode d'emparellament és el FLANN, ja que presenta per a tots una exactitud màxima (pràcticament igual que BF) i un temps mínim per la majoria d'imatges. D'ara endavant, es seguirà l'estudi tenint en compte només els emparellaments obtinguts amb FLANN, i els temps corresponents.

3.3.2. Suavitzat gaussià

Es començarà analitzant el comportament dels diferents algorismes de detecció i descripció enfront al suavitzat gaussià. A la següent figura es representen el nombre de punts característics extrets de les imatges suavitzades i la repetibilitat d'aquests amb la imatge inicial amb la finalitat d'avaluar els detectors.

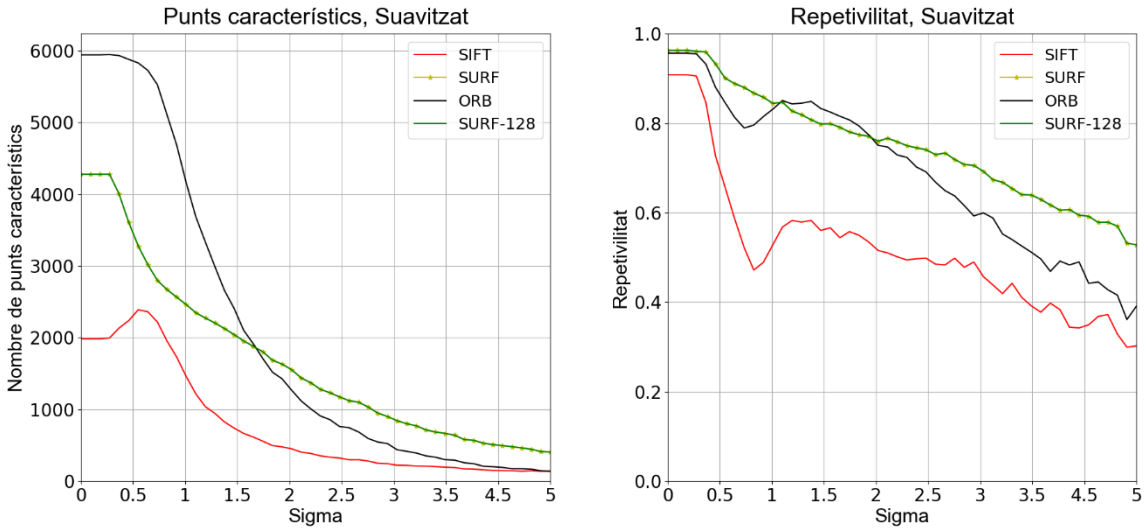


Figura 3.3.5. Punts característics de la imatge transformada i repetibilitat dels diferents algorismes de detecció. Transformació: Suavitzat gaussià. (Font: Elaboració Pròpia).

El detector que aparentment dona millor resposta és SURF. Tot i que per a tractar amb diferents escales de la imatge SURF utilitza l'aproximació de les gaussianes amb *box filters*, l'algorisme dona millors resultats en termes de punts extrets i repetibilitat d'aquests que SIFT. Els tests realitzats a diferents nivells de la piràmide que construeix ORB proporcionen també una alta repetibilitat, tot i que menor que amb SURF. SIFT queda en aquest cas en última posició clarament.

Els tres mètodes presenten una decaiguda de la repetibilitat al augmentar la dispersió de la gaussiana, però la decaiguda de SURF és la més constant.

Per a avaluar els descriptors es representen a la Figura 3.3.6 el nombre d'emparellaments, l'exactitud i el *recall*. S'observa que l'exactitud obtinguda amb els descriptors SIFT i SURF (tant estàndard com extens) és similar, mentre que el descriptor que utilitza ORB presenta una pendent de baixada lleugerament major.

Pel que fa a les correspondències emparellades, tal i com s'observa en el gràfic del *recall*, SIFT és clarament el millor mètode ja que està per sobre dels altres gairebé per tot el conjunt d'imatges (i sobretot a mesura que augmenta la dispersió de la gaussiana). ORB proporciona també un bon *recall*, clarament molt inferior que SIFT i amb pendent negativa. Cal destacar que la versió extensa de SURF emparella menys correspondències que l'estàndard en tot el conjunt i amb un valor molt inferior que amb els altres dos mètodes.

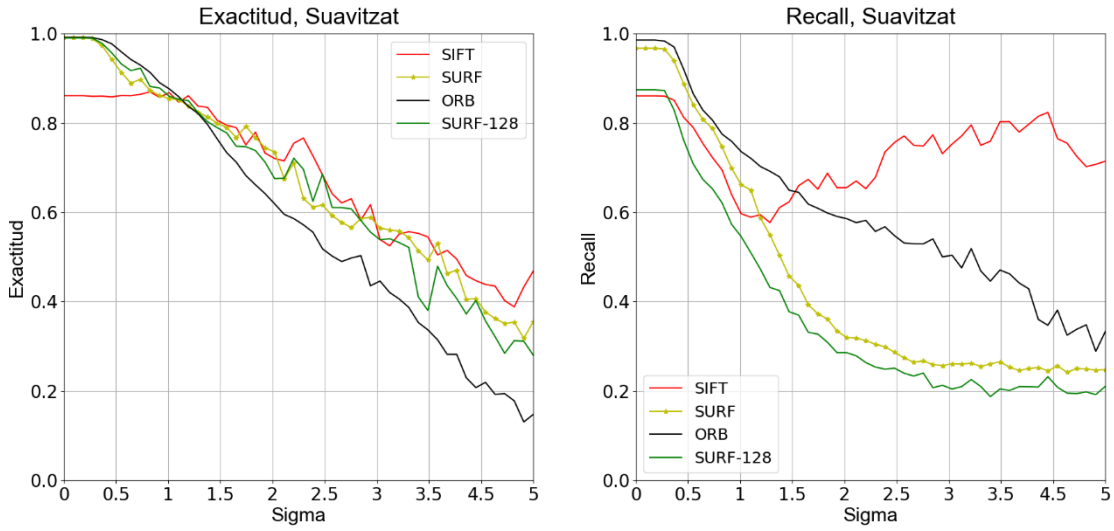


Figura 3.3.6. Nombre d'emparellaments, exactitud i *recall* dels diferents algorismes. Transformació: Suavitzat gaussià. (Font: Elaboració Pròpia).

Així doncs, es pot concloure que l'algorisme més invariant a grans rangs de difuminat gaussià és el SIFT. De mateixa manera, ORB també respon correctament a aquests canvis però a mesura que augmenten el seu rendiment disminueix. Finalment, remarcar que el rendiment de SURF disminueix molt ràpidament per a petits valors de dispersió (degut al descriptor, ja que és el mètode que obté una major repetibilitat) i que la versió extensa no aporta cap millora, sinó el contrari.

3.3.3. Rotació

La segona transformació a analitzar és la rotació. A la següent figura es representen el nombre de punts característics extrets de les imatges rotades *n* graus i la repetibilitat d'aquestes amb la imatge inicial.

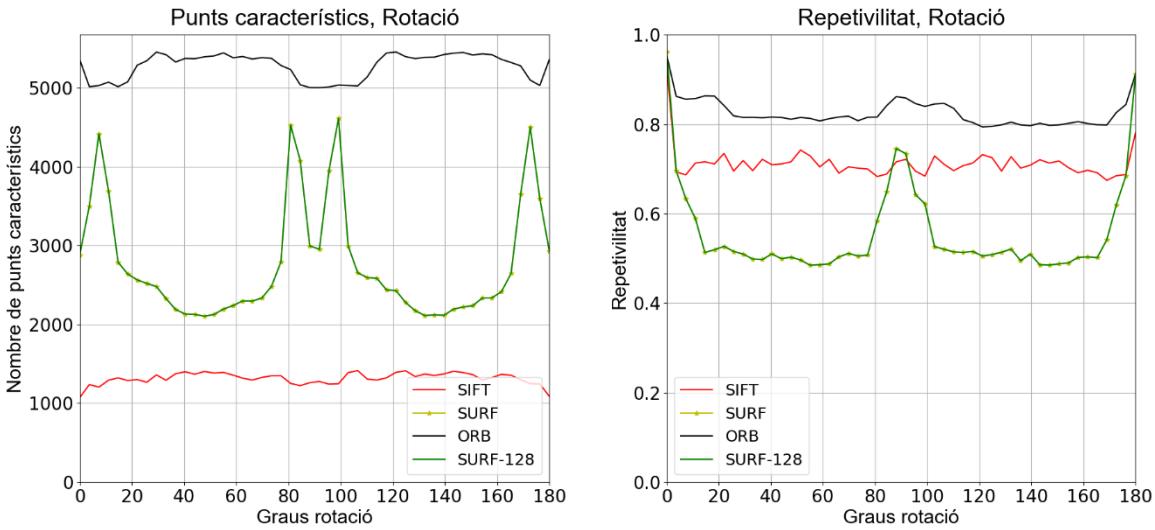


Figura 3.3.7. Punts característics de la imatge transformada i repetibilitat dels diferents algorismes de detecció. Transformació: Rotació. (Font: Elaboració Pròpia).

S'observa que el millor detector és, en aquest cas, el que utilitza ORB. Amb aquest, apart de detectar més punts, s'obté una repetibilitat major i més estable. SIFT també presenta bons resultats, amb una repetibilitat pròxima al 70% per a la majoria de rotacions.

Els detectors ORB i SIFT pateixen una disminució de la repetibilitat a partir d'aproximadament 5º graus de rotació i es mantenen estables fins als 175º, rotació amb la qual torna a augmentar la repetibilitat assolint un valor gairebé igual a l'inicial. SURF també presenta una resposta similar, però amb un clar màxim als 90º, i amb una repetibilitat menor en tot el conjunt. Aquest comportament l'expliquen la "simetria" dels *box filters* i *kernel Har wavelet* utilitzats en el detector SURF.

Com s'observa a les gràfiques de la Figura 3.3.8, el descriptor SURF de 128 dimensions no presenta cap millora significativa en comparació al de 64 dimensions i ambdós tenen un rendiment clarament inferior a ORB i SIFT.

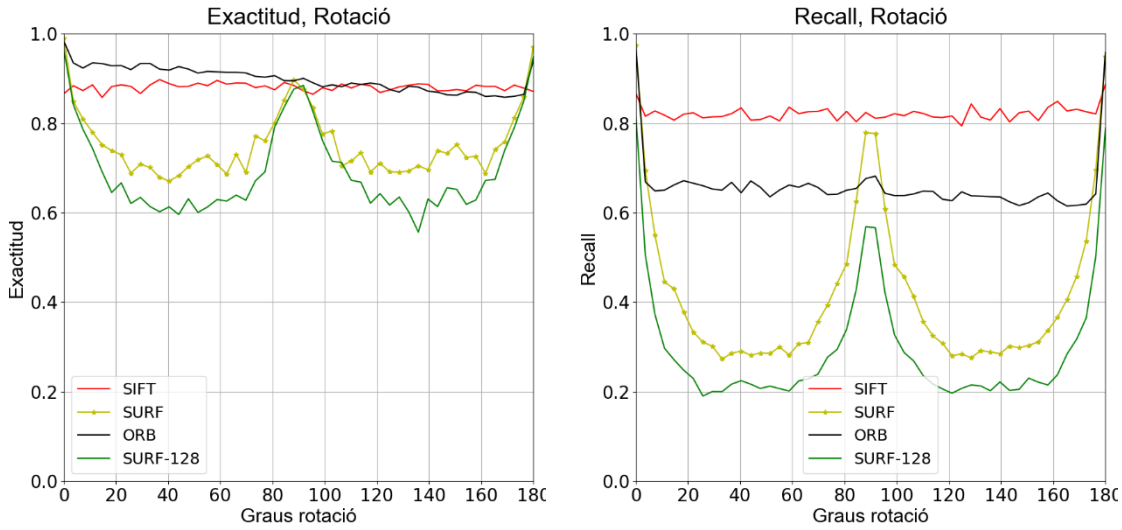


Figura 3.3.8. Nombre d'emparellaments, exactitud i *recall* dels diferents algorismes. Transformació: Rotació. (Font: Elaboració Pròpia).

Tant SIFT com ORB obtenen un alta exactitud en l'emparellament, però SIFT presenta una relació major entre el nombre d'emparellaments correctes i les correspondències. Cal remarcar que, tot i obtenir un *recall* major, el nombre de correspondències és molt major amb l'algorisme ORB, traduït en un nombre major de *true positives*.

3.3.4. Canvi d'escala

3.3.4.1. Escalat positiu

La següent transformació a analitzar és el canvi d'escala. Primerament s'analitzarà l'escalat positiu, obtingut augmentant la resolució de la imatge patró.

Com es pot veure a la Figura 3.3.9 el detector que presenta una millor repetibilitat i nombre de punts extrets és el que utilitza SIFT. El detector utilitzat en SURF presenta una repetibilitat similar a SIFT però sempre inferior a aquesta.

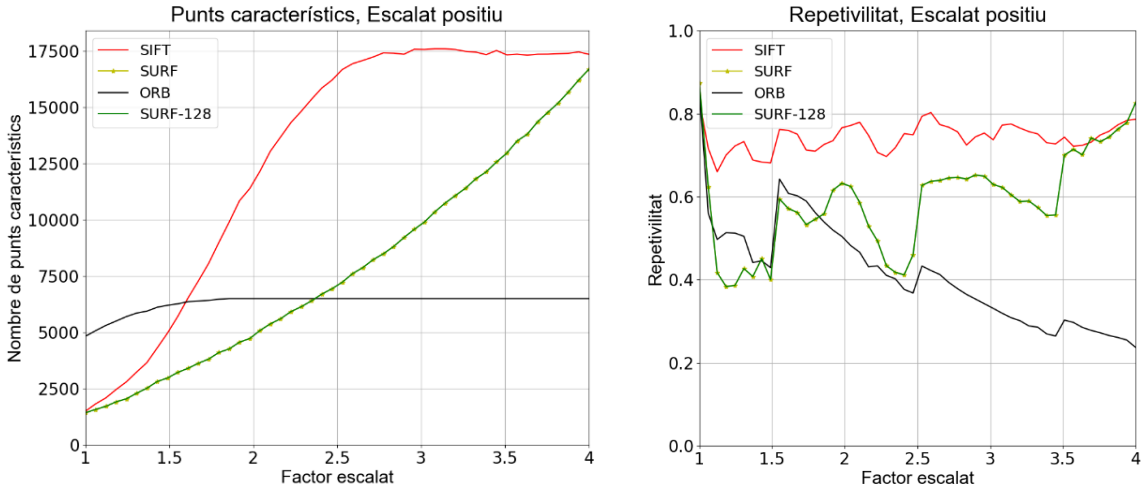


Figura 3.3.9. Punts característics de la imatge transformada i repetibilitat dels diferents algorismes de detecció. Transformació: Canvi d'escala positiu. (Font: Elaboració Pròpia).

També s'observa que a mesura que s'augmenta l'escalat els detectors SIFT i SURF es mantenen dins d'uns límits mentre que ORB té una clara decaiguda.

Pel que fa als descriptors i el seus emparellaments, tant en l'exactitud d'ells com la relació entre els *true positives* obtinguts amb les correspondències, el mètode més robust és una altre vegada SIFT. El segon mètode amb major robustesa és la versió estàndard de SURF, la qual obté un *recall* significativament major a la versió estàndard (tot i que una exactitud lleugerament inferior). En última posició queda ORB, amb totes les propietats menors que la resta d'algorismes.

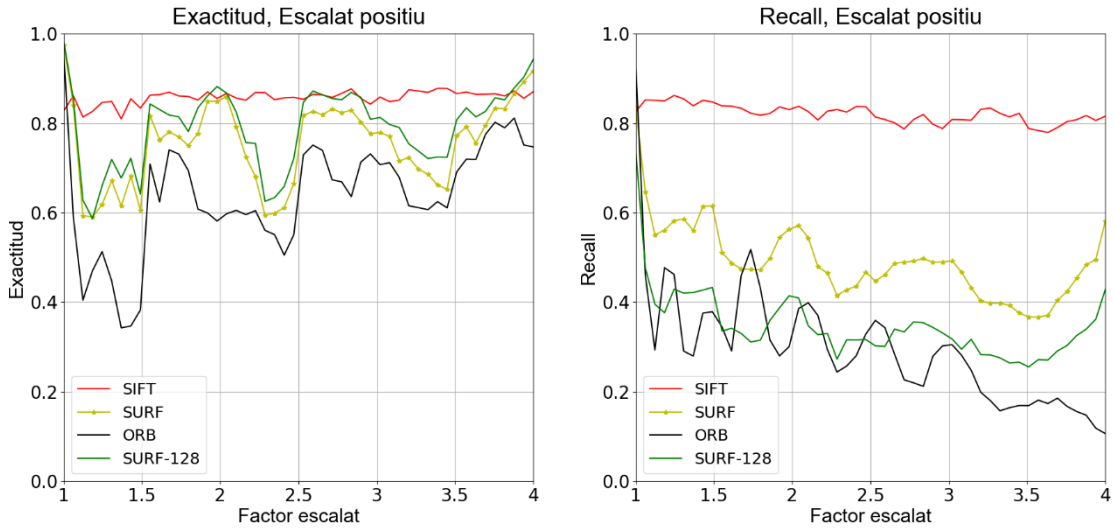


Figura 3.3.10. Nombre d'emparellaments, exactitud i *recall* dels diferents algorismes. Transformació: Canvi d'escala positiu. (Font: Elaboració Pròpia).

3.3.4.2. Escalat negatiu

Seguidament s'analitzarà el canvi d'escala negatiu, obtingut reduint la resolució de la imatge. Com es veu en la Figura 3.3.11, de mateixa manera que en l'escalat positiu, el detector SIFT és el que presenta millors resultats, seguit de SURF i finalment ORB.

També s'observa que, com en l'escalat positiu, a mesura que es disminueix l'escalat els detectors SIFT i SURF es mantenen estables dins d'uns límits. Però en l'experiment fet es veu que ORB també es manté relativament estable, quan amb l'escalat positiu s'apreciava una clara decaiguda. Aquest fet és degut a que en l'escalat positiu el nombre de punts extrets quedava limitat pel màxim establert, mentre que pel negatiu aquest nombre estava per sota de tal límit.

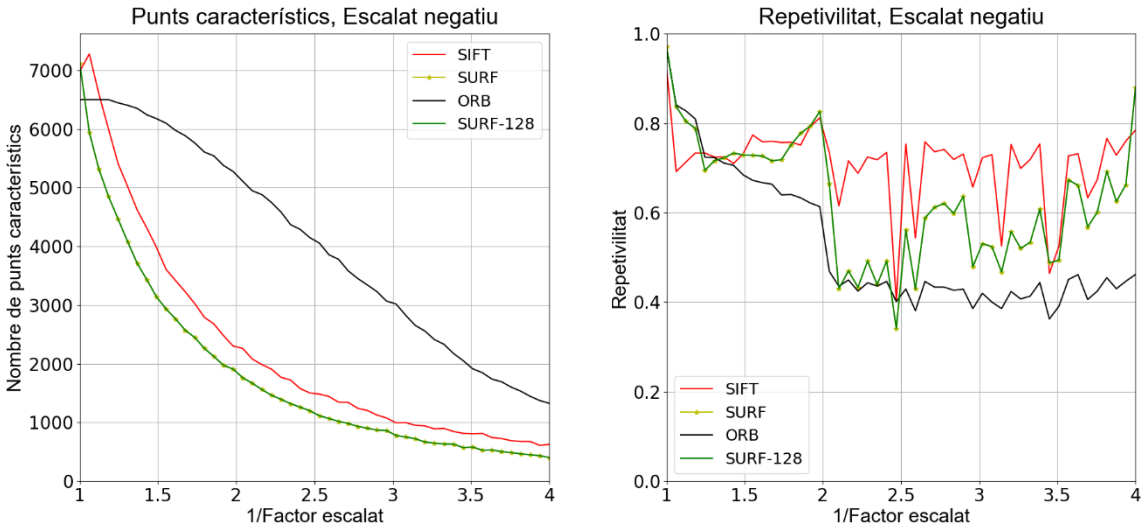


Figura 3.3.11. Punts característics de la imatge transformada i repetibilitat dels diferents algorismes de detecció. Transformació: Canvi d'escala negatiu. (Font: Elaboració Pròpia).

La robustesa dels descriptors és igual a la del escalat positiu, arribant a les mateixes conclusions que en l'apartat 3.3.4.1. Les representacions d'exactitud, *recall* i repetibilitat són relativament oscil·lants degut al submostreig de les imatges.

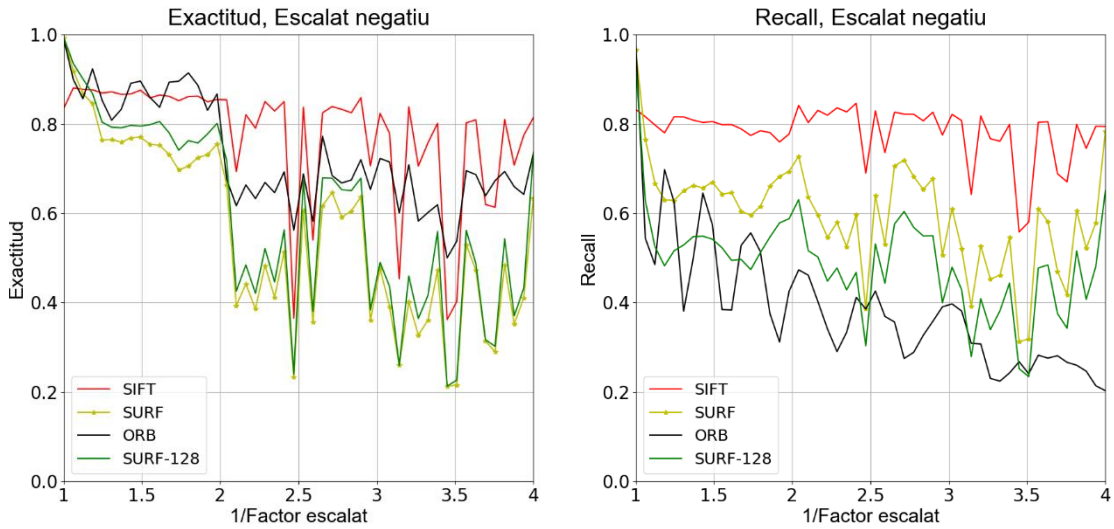


Figura 3.3.12. Nombre d'emparellaments, exactitud i *recall* dels diferents algorismes. Transformació: Canvi d'escala negatiu. (Font: Elaboració Pròpia).

3.3.5. Perspectiva

A la Figura 3.3.13 s'observa que el millor detector per a aquesta transformació és l'ORB. Amb aquest, apart de detectar més punts, s'obté la major repetibilitat. De nou, SIFT també presenta bons resultats, amb una repetibilitat semblant però lleugerament menor en tot el conjunt d'imatges. Es recorda que s'ha establert un límit màxim de punts per ORB i s'han retornat aquest màxim per a totes les imatges amb una gran repetibilitat, cosa que fa pensar que ORB seria capaç de detectar fins i tot més correspondències.

A mesura que augmenta el grau de transformació, la repetibilitat disminueix gairebé linealment pels tres detectors, tot i que amb un pendent major amb el detector SURF.

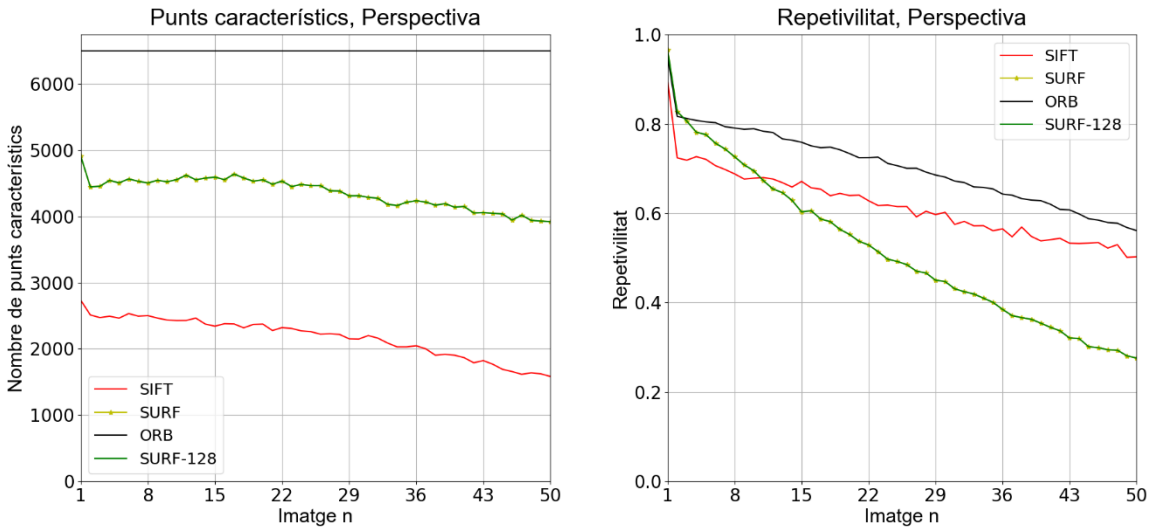


Figura 3.3.13. Punts característics de la imatge transformada i repetibilitat dels diferents algorismes de detecció. Transformació: Perspectiva. (Font: Elaboració Pròpia).

Els descriptors SIFT i ORB obtenen un alta exactitud i baixa decadència en l'emparellament, mentre que l'exactitud dels descriptors SURF inicialment en presenten una de major però que ràpidament decau i és superada per SIFT i ORB. Sota aquest tipus de transformacions, de nou, la versió extensa de SURF no presenta gaire cap millora en termes d'exactitud i obté un menor *recall.*

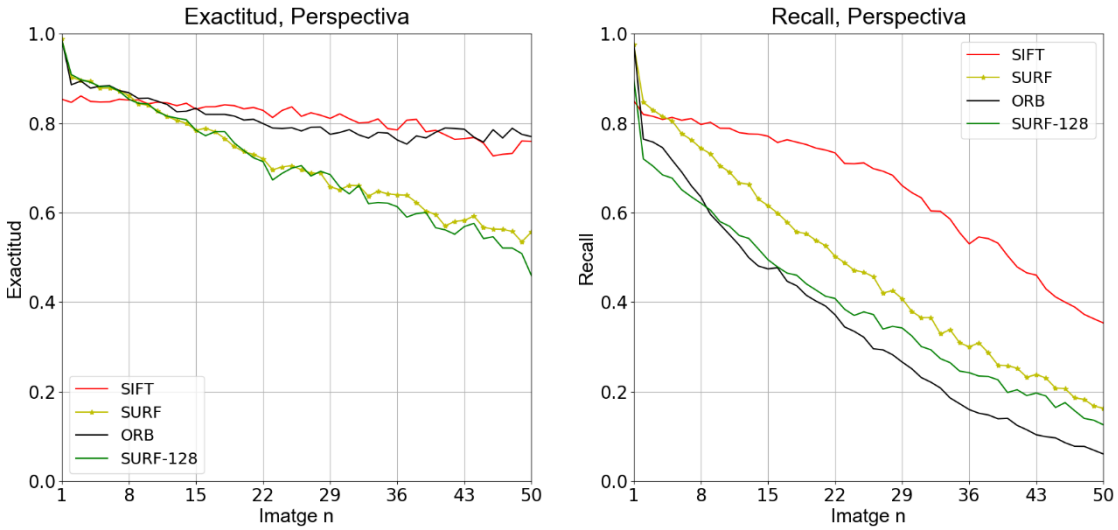


Figura 3.3.14. Nombre d'emparellaments, exactitud i *recall* dels diferents algorismes. Transformació: Perspectiva. (Font: Elaboració Pròpia).

ORB presenta el *recall* menor del conjunt, però al ser el mètode on clarament s'obtenen més correspondències, el nombre de *true positives* segueix sent molt elevat. Per tant, de nou ORB i SIFT són els mètodes que presenten major invariància a la transformació corresponent.

3.3.6. Lluminositat

3.3.6.1. Augment lluminositat

La última transformació a analitzar és la lluminositat, es començarà pels augments d'aquesta. Seguint amb el procediment d'anàlisi, en la següent figura es representen el nombre de punts característics i la repetibilitat d'aquests amb la imatge inicial.

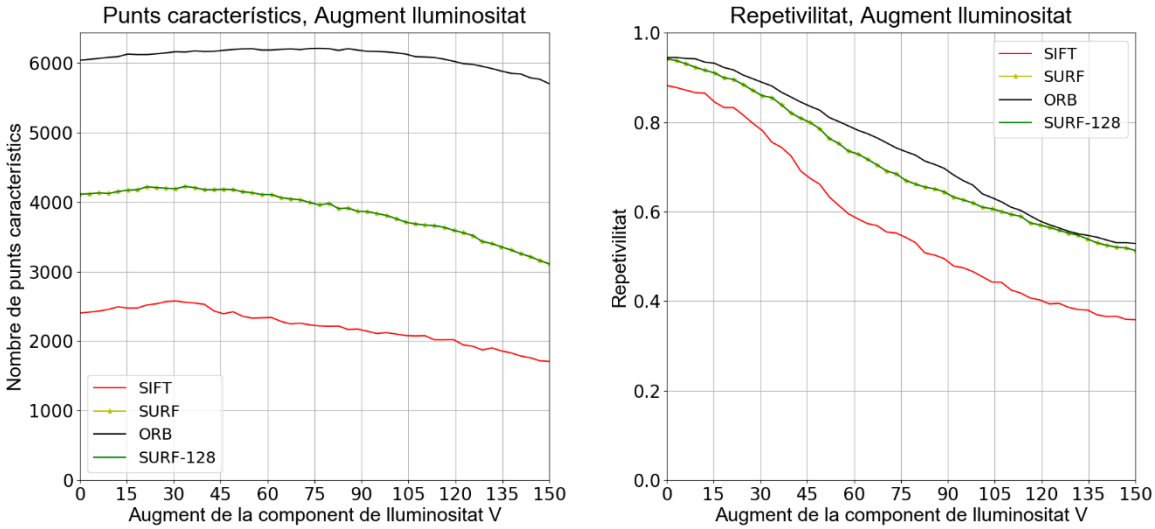


Figura 3.3.15. Punts característics de la imatge transformada i repetibilitat dels diferents algorismes de detecció. Transformació: Augment lluminositat. (Font: Elaboració Pròpia).

De nou el millor detector és, aparentment, l'ORB. Aquest obté una alta repetibilitat tot i detectar molts més punts d'interès que els altres algorismes. SURF presenta també una alta repetibilitat, gairebé igual a la de ORB. El darrer algorisme, SIFT, apart de detectar menys punts que els altres dos, també obté una repetibilitat menor. Els tres mètodes segueixen una trajectòria similar (amb diferents offsets entre elles) a mesura que les imatges pateixen l'augment gradual de la lluminositat.

Les exactituds obtingudes amb els descriptors SURF i ORB són les més elevades però amb lleugera tendència a la baixa (més accentuada amb els SURF). Mentre que la de SIFT és lleugerament inferior però pràcticament constant per tot el conjunt d'imatges. Observant el *recall* i el nombre d'emparellaments es pot concloure que tant SURF com ORB presenten millors resultats que SIFT per a canvis lleugers d'il·luminació, però que quan aquest canvi és més accentuat, on probablement hi existeixen canvis no lineals, SIFT obté un major nombre de correspondències emparellades, traduït en un nombre major de *true positives*.

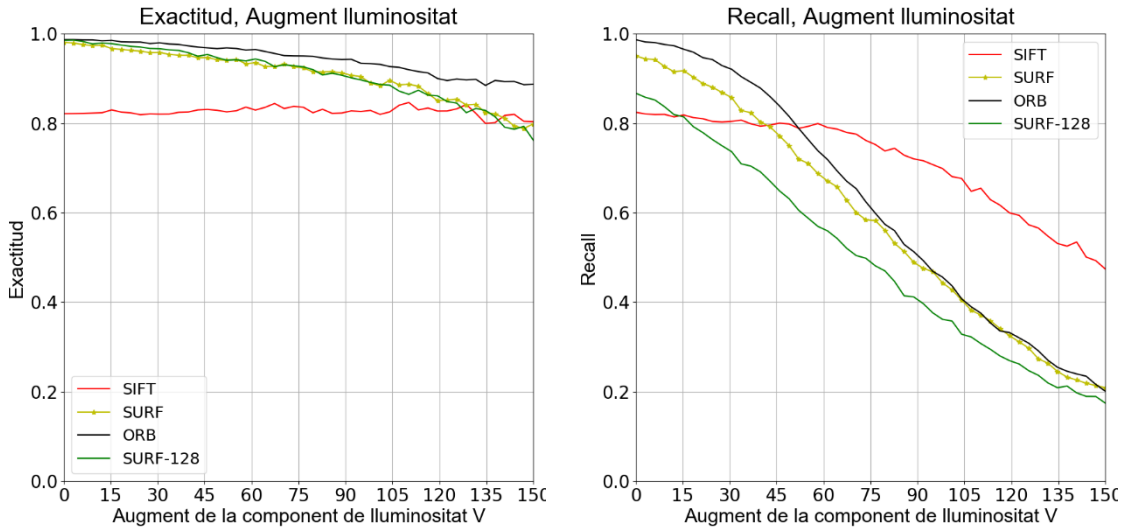


Figura 3.3.16. Nombre d'emparellaments, exactitud i *recall* dels diferents algorismes. Transformació: Augment Iluminositat. (Font: Elaboració Pròpia).

Cal remarcar també la millora considerable del rendiment de l'algorisme SURF al utilitzar el descriptors de 64 dimensions en comparació al de 128 dimensions.

3.3.6.2. Disminució Iluminositat

Finalment, s'analitza la disminució de la Iluminositat. Com era d'esperar, tant els resultats com les conclusions són pràcticament igual a l'augment de la Iluminositat.

Com es veu a continuació, el detector ORB és el que presenta millors resultats, seguit de SURF i finalment de SIFT.

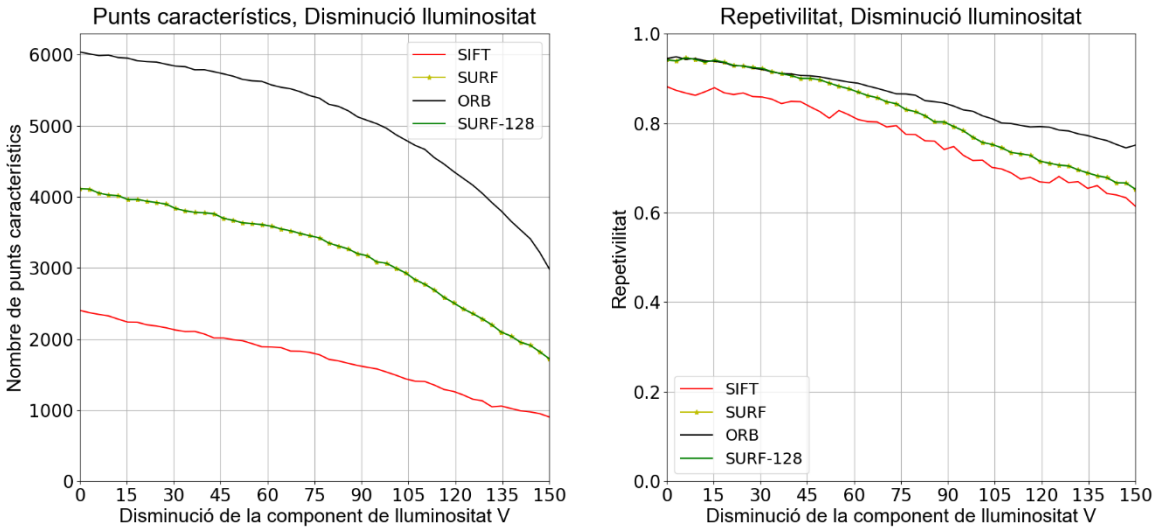


Figura 3.3.17. Punts característics de la imatge transformada i repetibilitat dels diferents algorismes de detecció. Transformació: Disminució lluminositat. (Font: Elaboració Pròpia).

Tant els valors com les trajectòries de l'exactitud i *recall* són també semblants als obtinguts en l'augment de lluminositat, com es veu a continuació:

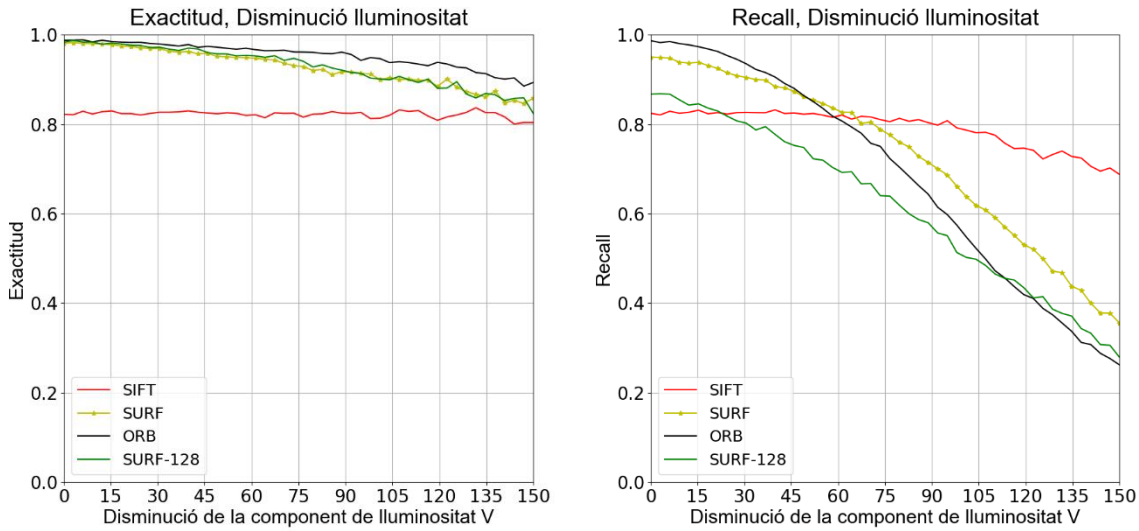


Figura 3.3.18. Nombre d'emparellaments, exactitud i *recall* dels diferents algorismes. Transformació: Disminució lluminositat. (Font: Elaboració Pròpia).

Per tant, les conclusions són les mateixes. Tant SURF com ORB responen adequadament a canvis lleugers d'il·luminació, mentre que SIFT és més constant i robust a canvis més forts.

3.3.7. Temps

Sorprenentment, tal i com es mostra en la Figura 3.3.19, el temps de detecció i descripció de l'algorisme SURF (tant la versió extensa com en l'estàndard) és major que en SIFT. S'observa també que la diferència de SURF al crear els descriptors de 64 dimensions o 128 és pràcticament nul.la. Clarament, ORB clarament és molt més ràpid que altres algorismes.

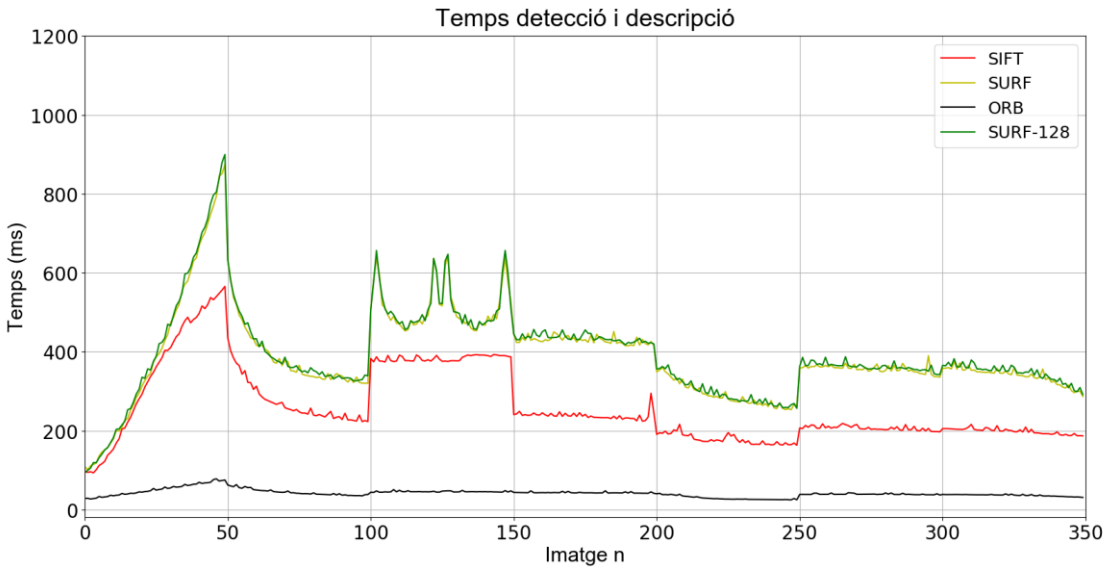


Figura 3.3.19. Temps de detecció i descripció de la imatge inicial i la imatge *n* del *data set*. (Font: Elaboració Pròpia).

Seguidament, s'il·lustren els temps d'emparellament de cada mètode i la seva rapidesa obtinguda a partir de la divisió entre el nombre de punts característics i el temps emprat.

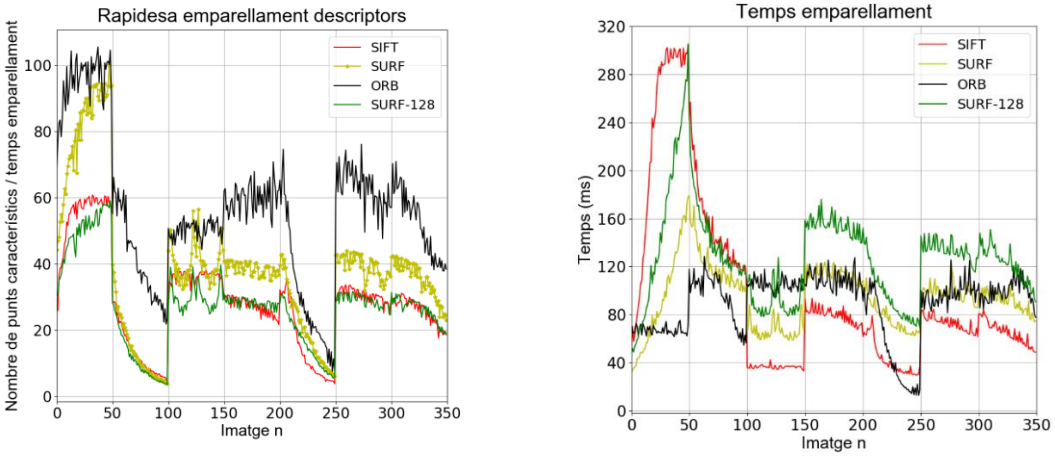


Figura 3.3.20. Rapidesa i temps d'emparellament dels diferents descriptors. (Font: Elaboració Pròpia).

Els descriptors ORB són els més ràpids d'emparellar ja que són binaris, els temps obtinguts experimentalment no són molt menors a la resta perquè, com s'ha vist anteriorment, s'extreuen molts més punts característics que amb els altres dos.

Els segons descriptors més ràpids són els SURF de 64 dimensions. Aquests, lògicament, són més ràpids que els de la seva versió extensa i que els de SIFT, ja que ambdós tenen 128 dimensions. Al tenir la mateixa dimensionalitat, els de SIFT i SURF extens tenen una velocitat molt semblant. En general el temps d'emparellament obtingut amb SIFT és menor a SURF, ja que SIFT obté menys punts característics generalment.

Finalment, es representa el temps de detecció descripció i emparellament dels diferents mètodes, juntament amb la velocitat amb què aquests extreuen *true positives* (obtinguda amb la divisió del nombre de *true positives* entre el temps de detecció, descripció i emparellament).

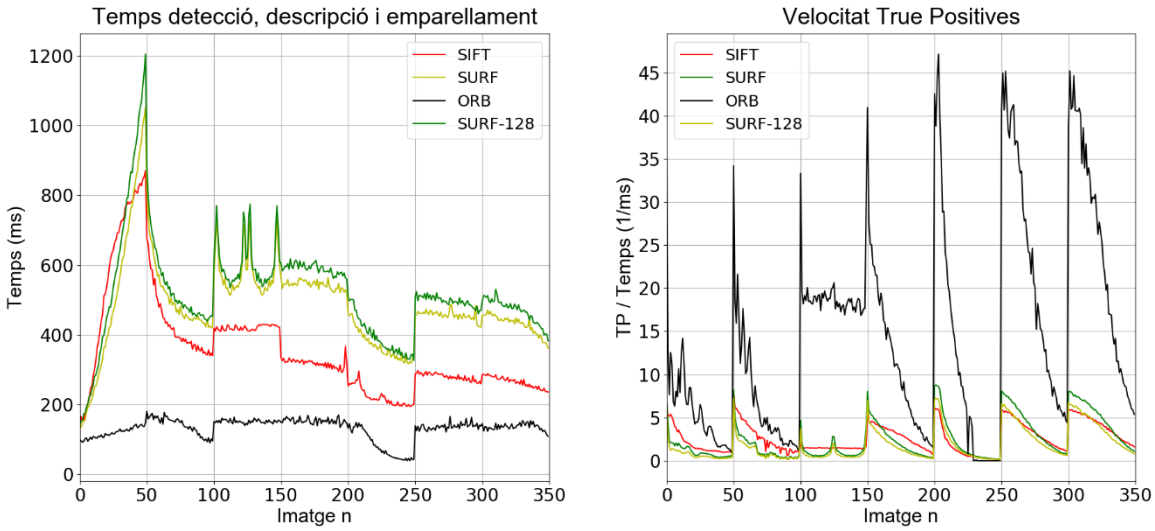


Figura 3.3.21. Temps de detecció, descripció i emparellament i velocitat de *true positives* dels diferents algorismes. (Font: Elaboració Pròpia).

Cal destacar també, la influència de la resolució de les imatges sobre el temps. Com es veu en les anteriors figures, els temps de detecció, descripció i emparellament augmenten significativament i gradualment en les primeres 50 imatges. Aquestes imatges són les del canvi d'escala positiu, obtingut al augmentar la resolució de les imatges.

Finalment es conclou afirmant que el mètode més ràpid és, amb diferència, ORB. Seguit de SIFT i finalment les dos versions de SURF.

3.3.8. Conclusions

Per a resumir les conclusions extretes de la comparació, les quals s'han anat realitzant amb més detall transformació per transformació, s'ha donat una puntuació numèrica a la robustesa de cada algorisme per a cada transformació i s'han recopilat a la Taula 4**.**

Les puntuacions corresponen a la mitjana dels valors mitjos de les propietats de més interès, aquestes són *recall*, exactitud i repetibilitat, per a cada transformació i dividida per deu per a aconseguir una puntuació amb rang entre 0 i 10. Per a més detall, a l'annex A1 s'han recopilat tots els valors mitjos utilitzats per a obtenir tal puntuació, és dir, els valors mitjos del *recall*, exactitud i repetibilitat per a cada algorisme i transformació.

Carl remarcar que a la columna del temps, s'hi introduirà la mitjana del temps de detecció, descripció i emparellament de tot el conjunt d'imatges. Finalment els resultats són els següents:

	Suavitzat gaussià	Rotació	Canvi d'escala	Perspectiva	Lluminositat	Temps
SIFT	6,4	8,0	7,8	7,0	7,5	355,55 ms
SURF	6,2	5,8	6,0	5,7	7,8	483,35 ms
SURF 128	6,0	5,2	4,9	5,4	7,5	522,06 ms
ORB	6,3	8,0	5,0	6,2	8,0	130,74 ms

Taula 4. Puntuacions dels diferents algorismes enfront les diferents transformacions i temps mitjà.

Es destaca que aquesta taula és útil per a veure per alt la invariància de cada algorisme enfront a les diferents transformacions, però per més detall sobre cadascuna es recomana prestar més atenció als apartats anteriors corresponents.

Es pot observar que l'algorisme SIFT obté una major puntuació que SURF per a gairebé totes les transformacions, a més, en la implementació de tals que proporciona la llibreria Open CV també és més ràpid. Així doncs, gairebé sempre serà més convenient utilitzar SIFT.

Per altra banda, s'observa que l'augment de la dimensionalitat del descriptor SURF no li aporta gairebé cap millora. Com s'ha vist als posteriors apartats, amb el descriptor extens augmenta lleugerament l'exactitud, però s'emparellen molts menys punts (ja que obté un *recall* sempre inferior a la versió estàndard), cosa que fa clarament més atractiva la versió estàndard de 64 dimensions, la qual es recorda que també és mes ràpida.

Finalment remarcar que amb ORB s'obtenen uns excel·lents resultats, i en aplicacions on el temps és clau, pot ser una bona alternativa a SIFT.

4. Aplicació: Reconeixement d'objectes

4.1. Metodologia utilitzada

En aquest apartat s'explica una metodologia pel reconeixement d'objectes, utilitzant els mètodes de detecció, descripció i emparellament estudiats en els apartats previs.

L'objectiu és, disposant d'una imatge de l'objecte el qual es desitja reconèixer, identificar-lo en una altre imatge o escena. Per exemple, a l'esquerra següent figura es mostra la imatge de l'objecte es qual es vol detectar i reconèixer, mentre que a la dreta es mostra l'escena on hi apareix i on posteriorment serà reconegut.

Figura 4.1.1. Esquerra: Objecte a reconèixer. Dreta: Escena on hi ha present l'objecte que es vol reconèixer. (Font: Elaboració Pròpia).

La metodologia emprada és la següent:

1. **Detecció i descripció:** S'aplica un algorisme de detecció o descripció a ambdós imatges. D'aquesta manera es té un conjunt de punts característics distintius de les dos imatges i els seus descriptors. Seguint amb l'exemple anterior, es mostren els punts característics en la següent imatge:

Figura 4.1.2. Representació dels punts detectats en l'objecte i l'escena. (Font: Elaboració Pròpia).

2. **Emparellament:** S'emparellen els descriptors d'ambdós imatges. Com s'ha vist anteriorment, el mètode d'emparellament *FLANN* seguit del rebuig en funció de la distància entre el primer i segon veí és el que dona millors resultats en relació a l'exactitud i el temps.

Figura 4.1.3. Representació dels emparellaments establerts entre els punts de l'objecte i l'escena. (Font: Elaboració Pròpia).

3. **Eliminació dels punts emparellats erròniament i càlcul d'homografia:** Com s'aprecia en la Figura 4.1.3, existeix un nombre important d'emparellaments erronis, anomenats falsos positius, que dificulten la detecció de l'objecte. Per a eliminar-los i calcular la matriu d'homografia s'utilitza l'algorisme RANSAC, qual s'explicarà en detall a l'apartat 4.1.1. Aquest procés només es realitza si s'ha trobat un mínim d'emparellaments, ja que es possible que l'objecte no estigui present a l'escena. Seguint amb l'exemple, el resultat és el següent:

Figura 4.1.4. Representació dels emparellaments establerts un cop s'han eliminat els emparellaments incorrectes. (Font: Elaboració Pròpia).

4. **Senyalització de l'objecte reconegut:** Un cop es disposa de la matriu de transformació entre el conjunt de localitzacions dels emparellaments de la primera i segona imatge, si el percentatge d'i*nliers* és major al 40%, es senyalitza l'objecte projectant els contorns de la primera imatge a l'escena, tal i com es veu a continuació:

Figura 4.1.5. Senyalització de l'objecte reconegut dins l'escena. (Font: Elaboració Pròpia).

4.1.1. Càlcul de la matriu d'homografia amb RANSAC

Primerament cal introduir RANSAC, un mètode iteratiu que permet ajustar els paràmetres d'un model a partir d'un conjunt de dades. Aquest es caracteritza per l'eliminació dels valors atípics, anomenats *outliers*, del conjunt de dades, obtenint així un ajust independent a aquests. A continuació es mostra una il·lustració de l'ajust d'una recta, on s'observa que els *outliers* (punts vermells) no influeixen en el resultat.

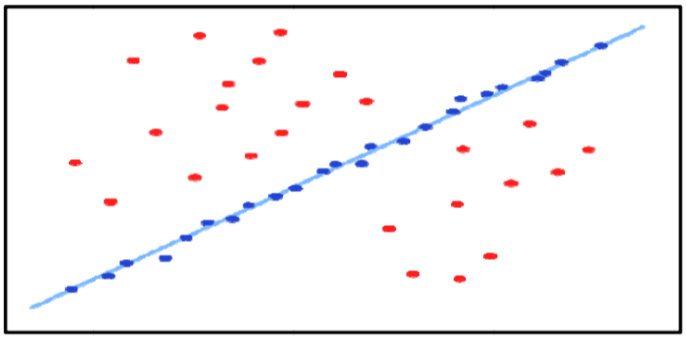


Figura 4.1.6. Ajustament d'una recta mitjançant RANSAC. (Font: [8]).

És un mètode determinista, ja que produeix un resultat raonable amb certa probabilitat, que augmenta amb el nombre d'iteracions. L'algorisme va ser publicar a l'article [8], el qual es convida a consultar si es desitja més informació.

S'utilitzarà RANSAC pel càlcul de la matriu d'homografia entre els punts p_i de l'objecte (imatge 1) emparellats amb els punts p_i' de l'objecte projectat a l'escena (part de la imatge 2). El conjunt de punts emparellats és anomenat P, i es recorda que no tots els punts han estat emparellats correctament.

També es recorda que per trobar la matriu d'homografia (transformació projectiva entre dos plans) es necessiten quatre parells de punts, és dir, quatre punts del primer pla i les seves 4 projeccions al segon. També que al ser una transformació projectiva, es treballa amb coordenades homogènies (introduïdes a l'apartat 3.1.4).

Seguidament s'exposa el funcionament de l'algorisme *RANSAC* adaptat per a calcular la matriu d'homografia i classificar els punts que s'han emparellat correctament com a *inliers* i els que s'han emparellat erròniament com a *outliers*. L'algorisme consta dels següents cinc passos:

1. Escollir aleatòriament un subconjunt de 4 parells de punts corresponents, p_i i p_i' , del conjunt P.
2. Calcular la matriu perspectiva H amb el subconjunt escollit. De manera que es compleixi (4.1.1) pels quatre parells de punts del subconjunt.

$$\begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} = H \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \tag{4.1.1}$$

3. Avaluar tots els punts segons $\tau(p_i, p_i)$. Com es veu en (4.1.2), (p_i, p_i) és un test que considera el parell de punts com a *outliers* si l'error en la re-projecció és major a un llindar ε, sinó els considera *inliers*.

$$\tau(p_i, p_i') = \begin{cases} outlier, & si \left(x_i' - \dfrac{h_{11}x_i + h_{12}y_i + h_{13}}{h_{31}x_i + h_{32}y_i + h_{33}}\right)^2 + \left(y_i' - \dfrac{h_{21}x_i + h_{22}y_i + h_{23}}{h_{31}x_i + h_{32}y_i + h_{33}}\right)^2 > \varepsilon \\ inlier, & Altrament \end{cases} \tag{4.1.2}$$

4. Tornar al pas 1 (N vegades).
5. Calcular la matriu perspectiva i classificar els punts del conjunt P com a *outliers* i *inliers* amb el millor subconjunt. El millor subconjunt és que té valor mínim de:

$$\sum_i \left(x_i' - \frac{h_{11}x_i + h_{12}y_i + h_{13}}{h_{31}x_i + h_{32}y_i + h_{33}}\right)^2 + \left(y_i' - \frac{h_{21}x_i + h_{22}y_i + h_{23}}{h_{31}x_i + h_{32}y_i + h_{33}}\right)^2 \tag{4.1.3}$$

Al aplicar aquest algorisme, és de gran importància escollir adequadament el nombre d'iteracions i el llindar ε.

El nombre d'iteracions N, s'ha d'assignar suficientment alt per assegurar amb la probabilitat p (normalment definida a 0,99) que com a mínim un subconjunt de punts aleatoris contingui 4 *inliers*. Sent u la probabilitat de que un punt del conjunt sigui *inlier* i $v = 1 - u$ la probabilitat d'observar un *outlier*, s'ha de complir el següent:

$$1 - p = (1 - u^4)^N \tag{4.1.4}$$

Operant amb (4.1.4) s'obté:

$$N = \frac{\log(1-p)}{\log(1-u^4)} \tag{4.1.5}$$

Partint de la premissa que com a mínim un 20 % del conjunt seran *inliers* s'obté:

$$N = \frac{\log(1-0{,}99)}{\log(1-0{,}2^4)} = 2875{,}98 \tag{4.1.6}$$

Finalment, arrodonint a l'alça s'ha escollit $N = 3000$. Es remarca que en funció de la rapidesa necessària en fer el reconeixement d'objectes es pot permetre augmentar o disminuir el nombre d'iteracions, tenint en compte el compromís entre rapidesa i probabilitat d'exactitud.

Pel que fa el valor del llindar ε, es recorda que l'error en la re-projecció és quadràtic. (veure la comparació expressada en l'equació (4.1.2). Així doncs, s'ha considerat 9 el valor del llindar. D'aquesta manera els punts amb error major a 9, corresponents a una distància de 9 píxels, seran classificats com a *outliers*.

4.2. Tolerància a la oclusió

En el reconeixement d'objectes, apart de la invariància a la rotació, perspectiva, lluminositat, escala, entre d'altres, és importat que els objectes puguin ser reconeguts tot i no estar completament presents en l'escena.

Per a aquest motiu, s'ha realitzat un experiment amb 4 tipus d'objectes (representats a la Figura 4.2.1). S'han escollit objectes amb diferent nombre de punts distintius per a analitzar també la dependència d'aquests. És immediat veure que el llibre és l'objecte més distintiu, seguit de l'ampolla de cava (gràcies a l'etiqueta), de l'encenedor i finalment de les ulleres.

Figura 4.2.1. Quatre diferents objectes utilitzats per determinar la tolerància a la oclusió. (Font: Elaboració pròpia).

Entrant en detall, es realitzaran dos tests per a cada objecte. Cada test està compost per 7 imatges amb graus d'oclusió incrementats gradualment. Com es veu a continuació, el primer test està compost per imatges amb diferents oclusions parcials (obtingudes al col·locar diferents objectes sobre el qual es vol reconèixer) i el segon està compost d'imatges on un objecte més gran es va col·locant damunt del que es vol reconèixer obstruint cada vegada una àrea major de tal. Seguidament es mostren les imatges de test del llibre, a l'annex A3 es poden consultar totes les imatges de ambdós tests per a cada objecte.

Figura 4.2.2. Totes les imatges de test del llibre. (Font: Elaboració pròpia)

S'aplicarà la metodologia explicada a l'apartat 4.1 pel reconeixement d'objectes amb els quatre algorismes de detecció i descripció estudiats (SIFT, SURF, SURF extens i ORB). Cada imatge de cada test es classificarà com a apte, semi-apte o apte, tal i com es veu a continuació:

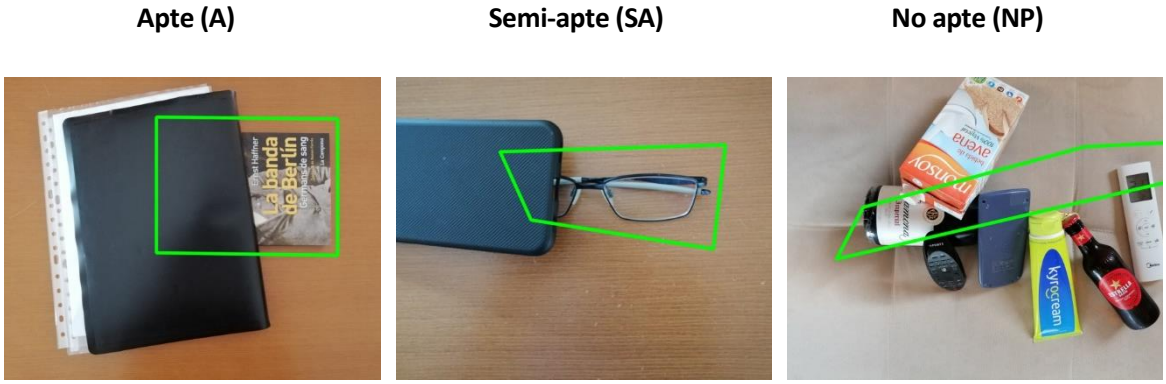


Figura 4.2.3. Classificació dels resultats de test. (Font: Elaboració pròpia).

A l'annex A4 es pot veure la taula construïda amb tots els resultats obtinguts. Aquesta s'ha simplificat i resumit en la següent:

	Aptes (%)	Semi aptes (%)	No aptes(%)
SIFT	70.83	8.33	20.83
SURF	52.08	6.25	41.67
SURF 128	54.17	10.42	35.42
ORB	41.67	6.25	52.08

Taula 5. Resum dels resultats dels diferents tests pels diferents objectes.

El millor mètode és, amb diferència, SIFT. Aquest és l'únic mètode que aconsegueix reconèixer adequadament les ulleres i l'encenedor amb un percentatge d'aptes major al 50 %.

La versió extensa de SURF dona una lleugera millora a l'estàndard a mesura que s'incrementa la oclusió. Ambdós mètodes però, tenen dificultats en emparellar objectes poc distintius com les ulleres.

En última posició queda ORB, el qual tolera l'oclusió d'objectes distintius de manera similar que SIFT, però la seva capacitat de reconeixement disminueix dràsticament per a objectes poc distintius com les ulleres i l'encenedor.

Per a més detall, es convida a veure l'annex A4**,** on hi ha recopilats els resultats de cada test.

4.3. Elaboració d'una interfície pel reconeixement en temps real d'objectes

Com el nom de l'apartat indica, s'ha elaborat una interfície pel reconeixement en temps real a través d'una *webcam* connectada a l'ordinador. Aquesta s'ha dissenyat amb la llibreria Tkinter, el paquet estàndard per elaborar interfícies gràfiques d'usuari (GUI) en Phyton.

La funcionalitat de la interfície o aplicació dissenyada és senzilla. L'usuari solament ha d'enfocar amb la càmera a l'objecte que es vulgui detectar, polsar el botó corresponent de la interfície per a capturar una imatge d'ell i immediatament s'efectua la metodologia explicada a l'apartat 4.1 pel reconeixement de l'objecte en qüestió. Així doncs, un cop s'ha capturat la fotografia de l'objecte aquest pot ser col·locat en qualsevol escena i l'aplicació l'intentarà reconèixer fins que es desitgi capturar i reconèixer un altre objecte.

Com es veu a la figura 4.3.2 es pot escollir amb quin algorisme de detecció i descripció es desitgi realitzar el reconeixement. De tal manera es pot comprovar i aprofundir en l'anàlisi que s'ha anat realitzant al llarg del treball.

En la següent figura s'exposa la principal funcionalitat o rutina que executa indefinidament l'aplicació.

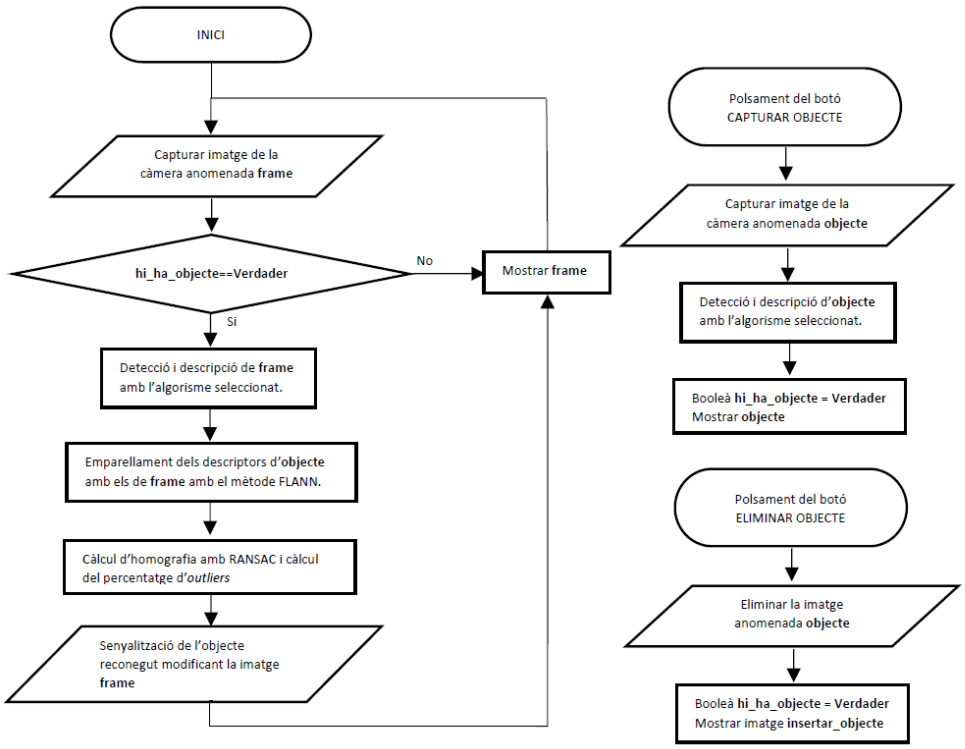


Figura 4.3.1. Diagrama de flux de la rutina principal de l'aplicació dissenyada. (Font: Elaboració Pròpia).

Apart del reconeixement d'objectes, a la interfície es poden visualitzar els diferents processos duts a terme com els punts detectats en ambdós imatges, els emparellaments i els emparellaments considerats *inliers*. També es mostra la freqüència amb que les fotos són mostrades, el nombre de punts detectats en cada imatge, el nombre d'emparellaments i el percentatge d'*inliers*.

L'última funcionalitat de l'aplicació és la possible modificació per part de l'usuari d'alguns paràmetres dels algorismes de detecció i descripció. Els paràmetres modificables es poden veure a la següent figura i es recorda que han estat descrits en detall a l'apartat 2.6.

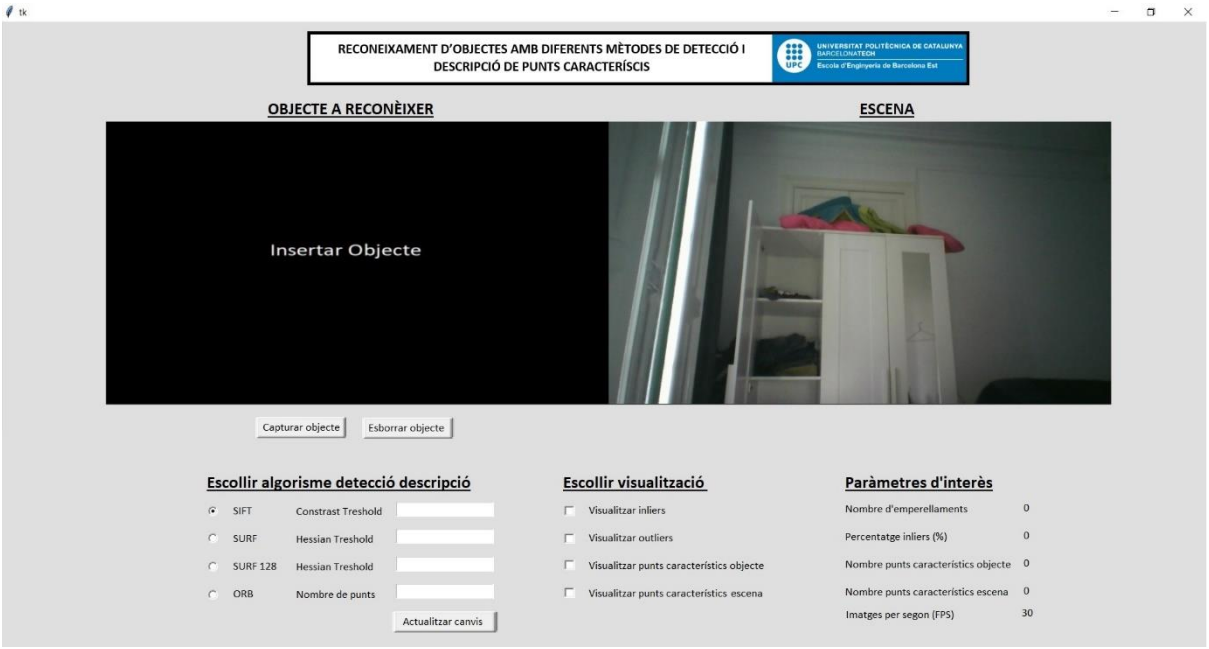


Figura 4.3.2. Aspecte inicial de la interfície dissenyada. (Font: Elaboració Pròpia).

L'anterior figura mostra la interfície inicial de l'aplicació dissenyada. Com s'ha dit anteriorment, l'usuari solament ha d'enfocar amb la càmera l'objecte que es desitja reconèixer, polsar el botó "Capturar objecte" i el reconeixement de l'objecte en qüestió es realitza immediatament. A continuació es pot veure un exemple del funcionament de l'aplicació.

Figura 4.3.3. Aspecte de la interfície dissenyada en funcionament. (Font: Elaboració Pròpia).

L'aplicació s'ha executat durant diversos minuts amb els paràmetres per defecte dels algorismes i s'ha calculat la mitjana de imatges per segon dels diferents algorismes. Els resultats són:

	SIFT	**SURF**	**SURF128**	**ORB**
Imatges per segon	11,74	11,53	10,83	19,3

Taula 6. Imatges per segon en l'aplicació dels diferents algorismes.

Convé remarcar que aquests valors són indicatius i útils per a veure la relació entre els algorismes i donar una idea aproximada de la velocitat dels algorismes i l'aplicació, però que són dependents de les prestacions de l'ordinador, de si l'objecte està o no present a l'escena, de les condicions de les imatges i els paràmetres dels algorismes, ja que un nombre major de punts característics comporta a un major temps d'emparellament .

Finalment es recorda que per a executar l'aplicació en qüestió es necessari tenir instal·lats Phyton, Open CV i Tkinter amb les versions especificades a la Taula 1.

5. Propostes de millora

En aquest apartat s'exposaran breument diferents propostes per a millorar l'estudi i aplicació realitzada. Cal remarcar que les hores que corresponen a els crèdits d'aquest treball fi de grau no són suficients per a elaborar aquestes propostes, però convé mencionar-les perquè aquestes són interessants. No s'entrarà en detall en cadascuna, sinó que s'enumeraran a mètode de resum de la següent manera:

- Afegir més algorismes en la comparació: S'han comparat els tres detectors/descriptors més utilitzats, estaria bé afegir-ne d'altres com BRISK.
- Augmentar el rang de transformacions, arribant a imatges amb transformacions més severes.
- Avaluació dels temps amb diferents ordinadors de diferents prestacions.
- Possibilitat de reconeixement de més objectes en la interfície: La metodologia que s'ha proposat pel reconeixement d'objectes és fàcilment escalable a més objectes.
- Possibilitat de reconèixer objectes presents amb imatges guardades a la memòria de l'ordinador.
- Mètodes de *tracking* en els objectes reconeguts: Un cop s'ha identificat un objecte, si en la pròxim imatge (recordem que l'aplicació és en temps real) s'explora primerament la regió veïna on l'objecte en qüestió quedava situat (on amb alta probabilitat estarà ubicat) es pot aconseguir un augment significatiu en la rapidesa de l'algorisme.

6. Anàlisi de l'impacte ambiental

L'avaluació de l'impacte ambiental és el conjunt d'estudis i anàlisis tècnics que permeten valorar els efectes que l'execució d'un determinat projecte pot causar sobre el medi ambient. Al tractar-se d'un estudi didàctic i d'una aplicació de *software* la qual pot ser utilitzada en una gran varietat de situacions, s'afirma que el projecte en qüestió no té cap repercussió directe en el medi ambient.

Tot i així, cal destacar que el potencial dels mètodes estudiats i de la visió per computador és enorme, i que les oportunitats que aquests ofereixen poden aplicar-se en un gran rang de situacions. Algunes d'aquestes són:

- Classificació de residus en reciclatge
- Detecció de residus al medi ambient
- Detecció d'incendis
- Salut i plagues en agricultura
- Manteniment urbà
- Estat del trànsit i aparcaments
- Seguiment de l'estat d'ecosistemes com esculls de corall

Conclusions

Per concloure el treball s'analitzaran tots els aspectes que s'han anat veien al llarg del treball per a determinar si els resultats han set els esperats i si s'han complert els objectius que es van proposar inicialment.

Començant pel principi, es creu que s'ha realitzat una documentació detallada i entenedora del funcionament dels algorismes i dels conceptes d'aquests. Ha set d'utilitat per, apart d'entendre els algorismes, poder aprofundir en la posterior comparativa.

En quant a la comparativa, cal remarcar que l'elaboració pròpia del conjunt d'imatges i del mètode d'extracció de propietats ha set satisfactori, aportant un conjunt de dades adequat per a realitzar amb èxit la comparació dels diferents algorismes.

Amb tal conjunt de dades s'ha pogut determinar que l'algorisme més robust és clarament SIFT i que la implementació de SURF que proporciona la llibreria "Open CV" no és el que s'esperava, ja que teòricament aquest es caracteritza per ser més ràpid que SIFT, amb certa pèrdua de qualitat. Així doncs, SIFT és clarament més atractiu que SURF, al ser majoritàriament més robust i ràpid.

També es conclou afirmant que ORB és una bona alternativa a SIFT en aplicacions on el temps és d'alta importància. La rapidesa d'aquest és notablement major a la dels anteriors, i amb una robustesa inferior però pròxima a SIFT.

Finalment, destacar que la metodologia pel reconeixement d'objectes utilitzada en l'aplicació ha donat bons resultats. Com s'ha vist, és capaç de reconèixer objectes amb cert grau d'oclusió i petits canvis de perspectiva. Convé destacar però, que aquesta manca en el reconeixement d'objectes poc distintius o amb grans graus de transformació.

Pressupost

Aquest apartat recull els costos implicats en la realització d'aquest projecte. Al treballar amb entorns de programació i llibreries *open source*, el pressupost s'ha desglossat solament en costos de recursos humans i amortitzacions. Convé destacar que no es tenen en compte costos com l'electricitat, espai i internet, ja que aquest treball ha estat realitzat principalment a la biblioteca de la universitat, la qual proporciona els recursos esmentats prèviament.

Pel que fa els recursos humans, s'ha realitzat la següent distribució de tasques:

- Recerca i elaboració de la documentació del funcionament dels diferents algorismes.
- Estudi comparatiu.
- Disseny i programació de l'aplicació.

En aquestes s'hi distingeixen clarament dos rols, el d'enginyer i el de programador. Seguidament s'exposen els costos en recursos humans.

Costos en recursos humans				
Descripció	**Rol**	**Preu/Hora (€/hora)**	**Hores**	**Preu (€)**
Recerca i elaboració de la documentació	Enginyer	20	150	3.000
Comparativa dels diferents algorismes	Enginyer	20	250	5.000
Disseny i programació de l'aplicació	Programador	15	200	3.000
		TOTAL	600	**11.000**

Taula 7. Costos en recursos humans

Com s'ha dit anteriorment, també s'han de tenir en compte els costos d'amortització dels recursos utilitzats. En l'elaboració d'aquest treball ha sigut necessari disposar d'un ordinador personal, valorat en 800 euros i una vida útil de 5 anys, el qual disposi del paquet d'*Office* personal, el qual té un preu de 69 euros l'any. Tenint en compte que la durada del projecte ha set de 6 mesos, els costos d'amortització són els següents:

Costos d'amortització					
Producte	**Temps (mesos)**	**Preu unitari (€)**	**Vida útil (anys)**	**Percentatge de dedicació (%)**	**Cost (€)**
Ordinador	6	800	5	90	72
Llicència office	6	69	1	75	25.9
				TOTAL	**97.9**

Taula 8. Costos d'amortització

Finalment, la suma dels dos anteriors proporciona el cost total del projecte, que és el següent:

Pressupost total	
Concepte	**Cost (€)**
Total recursos humans	11000
Total amortitzacions	97.9
Total	**11097.9**

Taula 9. Pressupost final del projecte

Bibliografia

[1] Lowe D. Distinctive image features from scale-invariant keypoints, International journal of computer vision, Vol. 60, Number 2, 2004, pp. 91-110.

[2] Bay H., Ess A., Tuytelaars T. and Van Gool L. Speeded-up robust features (SURF), Computer vision and image understanding, Vol. 110, Number 3, 2008, pp. 346-359.

[3] Rosten E. and Drummond T. Machine learning for highspeed corner detection, Computer Vision (ECCV 2006), Springer, 2006, pp. 430-443.

[4] Rublee E., Rabaud V., Konolige K. and Bradski G. ORB: an efficient alternative to SIFT or SURF, Computer Vision (ICCV), IEEE, 2011, pp. 2564-2571.

[5] Calonder M., Lepetit V., Strecha C. and Fua P. Brief: Binary robust independent elementary features, European Conference on Computer Vision, 2010.

[6] Szeliski R. Computer Vision: Algorithms and Applications, Washington, USA. Springer, 2010.

[7] Kole S., Agarwal C., Gupta T and Singh S. SURF and RANSAC: A Conglomerative Approach to Object Recognition, International Journal of Computer Applications, Vol. 109, Number 2, 2015 pp. 0975 – 8887.

[8] Fischer M. and Bolles R. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography, CACM, Vol. 24, Number 6, pp. 381–395, 1981.

[9] Fei F.L. (2011). Lecture 11: Detectors and descriptors. Stanford University Lab. Disponible a: "http://vision.stanford.edu/teaching/cs231a_autumn1112/lecture/lecture11_detectors_descriptors_cs231a.pdf".

Annexes

A1. Valors mitjans de propietats d'interès dels diferents algorismes per diferents transformacions

Repetibilitat (%)	SIFT	SURF	SURF 128	ORB
Suavitzat gaussià	52	75	75	69
Rotació	71	56	56	83
Canvi d'escala P	74	59	59	41
Canvi d'escala N	71	63	63	52
Perspectiva	62	52	52	70
A Lluminositat	58	71	71	74
D Lluminositat	77	82	82	86

Taula 10. Valors mitjans de repetibilitat pels diferents algorismes i transformacions.

Exactitud (%)	SIFT	SURF	SURF 128	ORB
Suavitzat gaussià	68	67	66	59
Rotació	88	76	70	90
Canvi d'escala P	86	76	81	65
Canvi d'escala N	78	56	6	75
Perspectiva	82	72	71	82
A Lluminositat	83	91	92	95
D Lluminositat	82	93	94	96

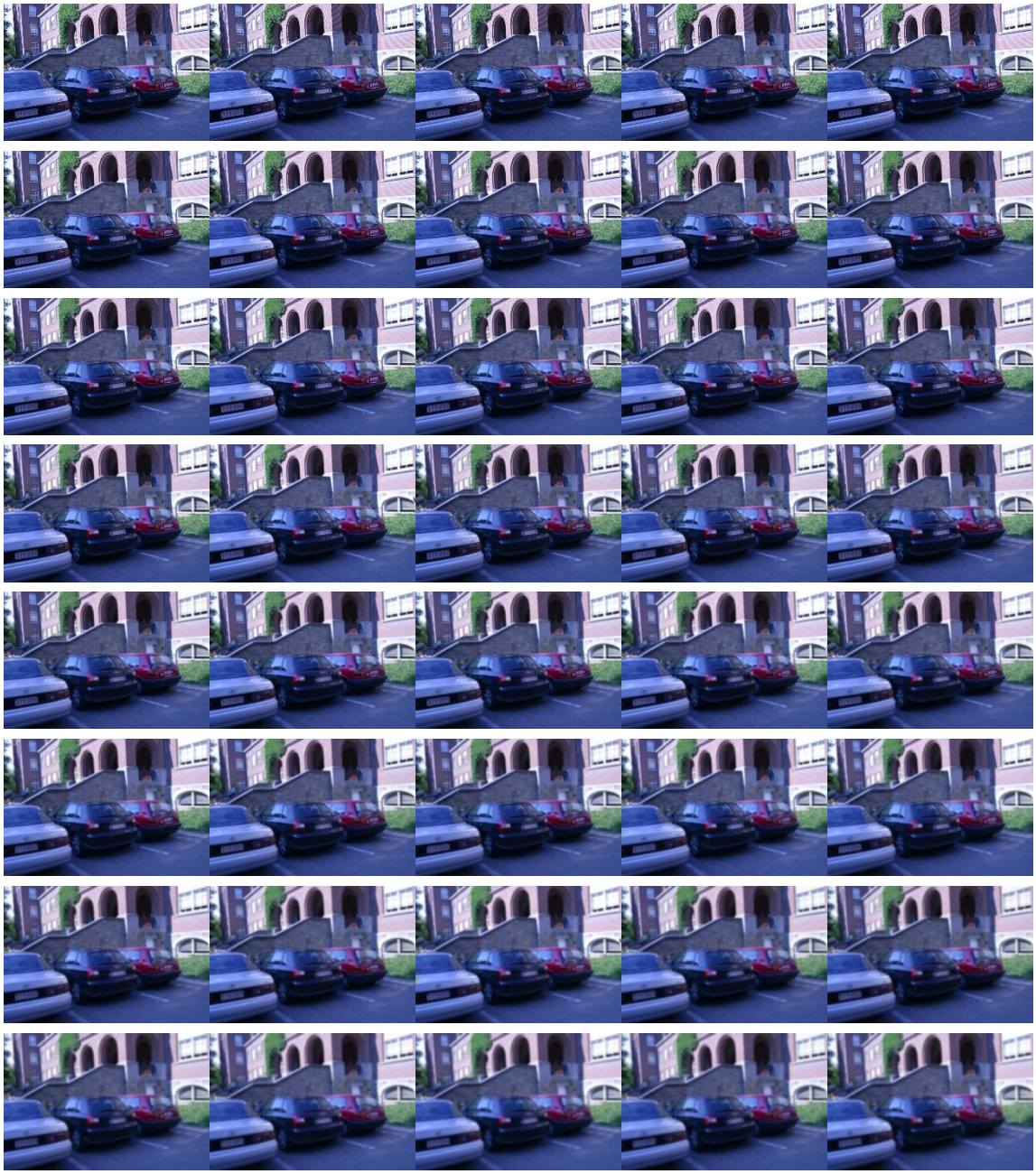
Taula 11. Valors mitjans d'exactitud pels diferents algorismes i transformacions.

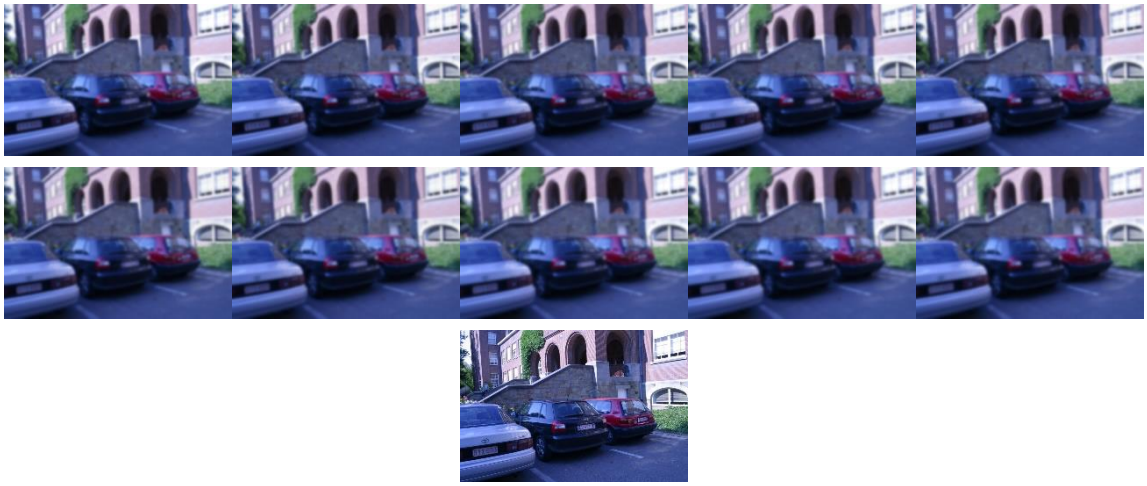
Recall (%)	SIFT	SURF	SURF 128	ORB
Suavitzat gaussià	73	45	38	60
Rotació	82	41	29	66
Canvi d'escala P	82	48	34	29
Canvi d'escala N	78	60	48	38
Perspectiva	65	47	39	34
A Lluminositat	71	58	49	61
D Lluminositat	79	73	61	68

Taula 12. Valors mitjans del *recall* pels diferents algorismes i transformacions.

A2. *Dataset* d'imatges utilitzades en la comparativa

A continuació es mostren les imatges del *data set*. Es mostraran amb una taula per a cada subconjunt, la primera imatge de cadascuna correspon a la imatge patró i la última a la imatge amb major transformació. Convé remarcar que per els subconjunts de imatges amb canvi d'escala positiu i negatiu no hi són presents ja que aquestes només han patit un canvi de resolució.

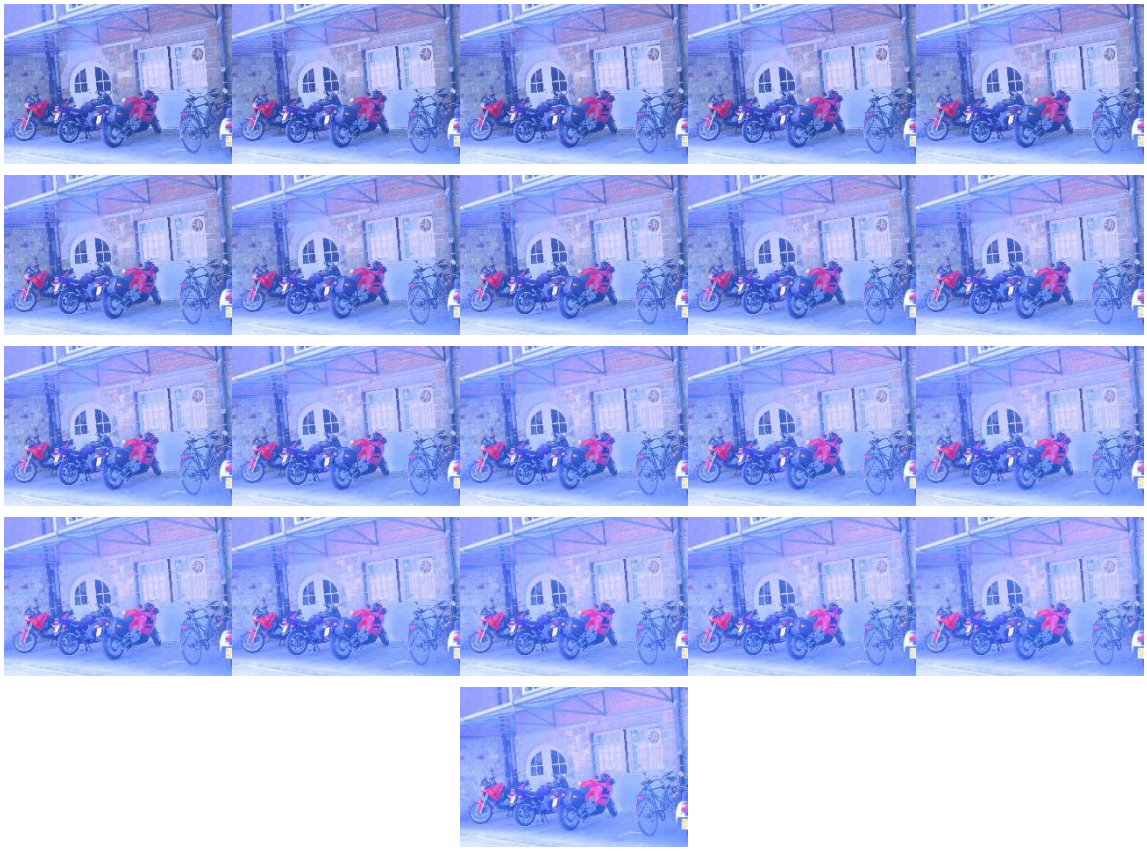




Taula 13. Conjunt d'imatges generades amb l'increment gradual de suavitzat gaussià

Taula 14. Conjunt d'imatges generades amb diferents rotacions.

Taula 15. Conjunt d'imatges generades amb un increment gradual de transformació perspectiva.



Taula 16. Conjunt d'imatges generades amb un increment gradual de lluminositat.

Taula 17. Conjunt d'imatges generades amb un decrement gradual de lluminositat.

A3. Conjunt d'imatges utilitzades per a determinar la tolerància a l'oclusió

<table>
<tr><th colspan="4">TEST 1 - Llibre</th></tr>
<tr><th>Imatge Patró</th><th>Imatge 1</th><th>Imatge 2</th><th>Imatge 3</th></tr>
<tr><td rowspan="2"> 
</td><td>  </td><td>  </td><td>  </td></tr>
<tr><th>Imatge 4</th><th>Imatge 5</th><th>Imatge 6</th></tr>
<tr><td>  </td><td>  </td><td>  </td></tr>
</table>

Taula 18. Imatges del test 1, objecte: Llibre.

TEST 2 - Llibre			
Imatge Patró	**Imatge 1**	**Imatge 2**	**Imatge 3**
			
	Imatge 4	**Imatge 5**	**Imatge 6**
			

Taula 19. Imatges del test 2, objecte: Llibre.

TEST 1 - Encenedor			
Imatge Patró	**Imatge 1**	**Imatge 2**	**Imatge 3**
			
	Imatge 4	**Imatge 5**	**Imatge 6**
			

Taula 20. Imatges del test 1, objecte: Encenedor.

TEST 2 - Encenedor			
Imatge Patró	**Imatge 1**	**Imatge 2**	**Imatge 3**
			
	Imatge 4	**Imatge 5**	**Imatge 6**
			

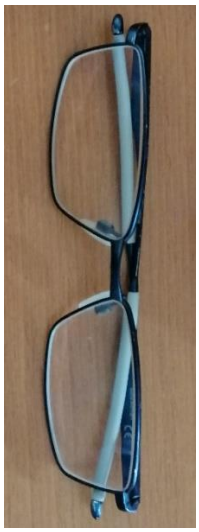
Taula 21. Imatges del test 2, objecte: Encenedor.

TEST 1 - Ampolla			
Imatge Patró	**Imatge 1**	**Imatge 2**	**Imatge 3**
			
	Imatge 4	**Imatge 5**	**Imatge 6**
			

Taula 22. Imatges del test 1, objecte: Ampolla.

TEST 2 - Ampolla			
Imatge Patró	**Imatge 1**	**Imatge 2**	**Imatge 3**
			
	Imatge 4	**Imatge 5**	**Imatge 6**
			

Taula 23. Imatges del test 2, objecte: Ampolla.

TEST 1 - Ulleres			
Imatge Patró	**Imatge 1**	**Imatge 2**	**Imatge 3**
			
	Imatge 4	**Imatge 5**	**Imatge 6**
			

Taula 24. Imatges del test 1, objecte: Ulleres.

TEST 2 - Ulleres			
Imatge Patró	**Imatge 1**	**Imatge 2**	**Imatge 3**
			
	Imatge 4	**Imatge 5**	**Imatge 6**
			

Taula 25. Imatges del test 2, objecte: Ulleres.

A4. Taula dels resultats dels tests d'oclusió

Objecte	Algorisme	TEST	Img1	Img2	Img3	Img4	Img5	Img6
Llibre	SIFT	T1	A	A	A	A	A	NA
		T2	A	A	A	A	A	SA
	SURF	T1	A	A	A	A	NA	NA
		T2	A	A	A	A	A	NA
	SURF128	T1	A	A	A	A	SA	NA
		T2	A	A	A	A	A	NA
	ORB	T1	A	A	A	A	NA	NA
		T2	A	A	A	A	A	NA
Ampolla	SIFT	T1	A	A	A	A	SA	NA
		T2	A	A	A	A	A	NA
	SURF	T1	A	A	A	SA	NA	NA
		T2	A	A	A	A	A	NA
	SURF128	T1	A	A	A	SA	NA	NA
		T2	A	A	A	A	A	NA
	ORB	T1	A	A	A	A	SA	NA
		T2	A	A	A	A	SA	NA
Encenedor	SIFT	T1	A	A	A	A	A	NA
		T2	A	A	A	A	A	NA
	SURF	T1	A	A	NA	NA	SA	NA
		T2	NA	A	A	NA	NA	NA
	SURF128	T1	A	A	NA	NA	SA	NA
		T2	A	A	A	A	NA	NA

	ORB	**T1**	NA	NA	NA	NA	NA	NA
		T2	NA	A	A	NA	SA	NA
Ulleres	**SIFT**	**T1**	A	A	SA	SA	NA	NA
		T2	A	A	A	NA	NA	NA
	SURF	**T1**	A	A	NA	NA	NA	NA
		T2	A	A	SA	NA	NA	NA
	SURF128	**T1**	A	SA	NA	NA	NA	NA
		T2	A	A	SA	NA	NA	NA
	ORB	**T1**	A	NA	NA	NA	NA	NA
		T2	NA	NA	NA	NA	NA	NA

Taula 26. Resultats complerts dels tests realitzats per determinar la tolerància a l'oclusió. A = apte, SA = semi-apte, NA= no apte.